МІНІСТЭРСТВА АДУКАЦЫІ РЭСПУБЛІКІ БЕЛАРУСЬ
БЕЛАРУСКІ ДЗЯРЖАЎНЫ ТЭХНАЛАГІЧНЫ УНІВЕРСІТЭТ

УРБАНОВІЧ П.П.

АРЫФМЕТЫЧНЫЯ І ЛАГІЧНЫЯ АСНОВЫ ЛІЧБАВЫХ АЎТАМАТАЎ

Вучэбны дапаможнік па аднайменнаму курсу
для студэнтаў спецыяльнасці Т.14.02 «Паліграфічнае абсталяванне і сродкі апрацоўкі інфармацыі»

Мінск 1999

УДК 681.322

Разгледжаны і рэкамендаваны да выдання рэдакцыйна-выдавецкай радай універсітэта

Урбановіч П.П. Арыфметычныя і лагічныя асновы лічбавых аўтаматаў. Вучэбны дапаможнік па аднайменнаму курсу для студэнтаў спецыяльнасці Т.14.02 «Паліграфічнае абсталяванне і сродкі апрацоўкі інфармацыі». - Мн.: БДТУ, 1999.

.

Рэцэнзенты: прафесар БДЭУ Маразевіч А.М.,
дацэнт БДУІР Герман А.В.

Па тэматычным плане выданняў вучэбна-метадычнай літаратуры універсітэта на 1999 г., паз. 114

© Беларускі дзяржаўны тэхналагічны універсітэт, 1999.

© Складанне. П.П.Урбановіч, 1999.

УВОДЗІНЫ

Сярод разнастайных электронна-вылічальных сродкаў вылучаюцца два вялікія класы вылічальных машын – аналагавы (**АВМ**) і лічбавыя (**ЛВМ**). У АВМ усе лікі (ўваходныя, прамежкавыя і выхадныя даныя) падаюцца ў выглядзе фізічных працэсаў, якія характарызуюцца сваёй велічынёй. Лічбы ў АВМ могуць быць пададзены, напрыклад, ў выглядзе механічных перамяшчэнняў, электрычных напружанняў і г.д. У ЛВМ усе лікі адлюстроўваюцца ў выглядзе сукупнасці лічбаў у пазіцыйнай сістэме лічэння.

Найбольшае распаўсюджанне атрымала двайковая сістэма лічэння (**СЛ**), у якой кожная лічба ліку можа мець значэнне нуля або адзінкі. ЛВМ часта называюць электроннымі ЛВМ (**ЭЛВМ**) або проста ЭВМ (зараз часцей ўжывальным з’яўляецца тэрмін *прафесійная персанальная ЭВМ*, **ППЭВМ** або проста **ПК**). Паколькі АВМ прадстаўляюць сабой асобны клас устройстваў і з’яўляюцца зместам другіх вучэбных дысцыплін, у дадзеным дапаможніку, прызначаным для вывучэння асноў пабудовы і функцыянавання устройстваў і сістэм на падставе лічбавых сігналаў, разглядаюцца толькі **ЭЛВМ**.

У агульным плане ЭЛВМ складаецца з трох асноўных блокаў (устройстваў). Гэта арыфметыка-лагічнае (*АЛУ*), аператыўнае запамінальнае (*АЗУ*) і устройства кіравання (*УК*). Акрамя таго, ў склад машыны ўваходзяць адно або некалькі устройстваў уводу даных і вываду вынікаў, другія знешнія устройствы (*ЗУ*). Такім чынам, з улікам таго, што студэнты асобна вывучаюць дысцыпліны “Знешнія устройствы ЭВМ” і “ЭВМ і вылічальныя сістэмы”, уздакладнім прадмет нашага разгляду: арыфметычныя і лагічныя устройствы ЭЛВМ.

1. Сістэмы лічэння і формы ўяўлення лічбаў у ЭВМ і ПК

1.1. Агульныя ўласцівасці пазіцыйнай сістэмы лічэння

Сістэмы лічэння (СЛ) падзяляюцца на пазіцыйныя і непазіцыйныя. У ЭВМ і ПК выкарыстоўваюцца пазіцыйныя сістэмы – гэта сістэмы запісу розных па велічыне лікаў абмежаванай колькасцю сімвалаў, прычым ранг кожнай лічбы залежыць ад яе пазыцыі ў гэтым ліку. Колькасць сімвалаў, абмежаваная некаторай велічынёй **A,** называецца асновай СЛ. Калі **A** =10 (0,1,…,9), то сістэма – дзесятковая, калі **A** = 2 (0,1), то – двайковая і г.д. У агульным выпадку змешаны лік **X** можна запісаць у такой форме:

$$X = \sum_{i=-f}^{n-1} x_i \cdot A^i$$

дзе **n** - агульная колькасць разрадаў цэлай часткі (**n**-ы разрад будзем лічыць старшым); **f** - агульная колькасць разрадаў дробнай часткі (**f** – малодшы); $\mathbf{x_i}$ – лічба ў **i**-тым разрадзе; **A** – аснова СЛ. Такім чынам,

$$\mathbf{X = x_n*A^{n-1} + x_{n-1}* A^{n-2} +\ldots+ x_1* A^0 + x_0*A^{-1} + \ldots + x_{-f+1}*A^{-f}.}$$

Напрыклад: $115.275=1\cdot10^2+1\cdot10^1+5\cdot10^0+2\cdot10^{-1}+7\cdot10^{-2}+5\cdot10^{-3}$.

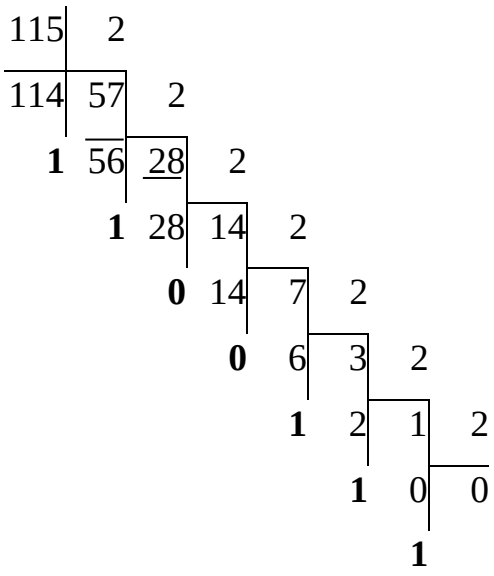
Аналіз паказвае існаванне асноўных прынцыпаў выкарыстання СЛ:

1) мінімальны лік, які разглядаецца ў ЭВМ, ёсць $\mathbf{A^{-f}}$ (шаг вылічэнняў – адзінка самага малодшага разраду), 2) максімальнаы лік – $\mathbf{A^n-1}$, 3) агульная колькасць лікаў, якія перапрацоўваюцца ЭВМ у дадзенай сістэме – $\mathbf{A^{n+f}}$.

1.2.Перавод цэлых лікаў з адной СЛ у другую

Для пераводу цэлага ліку неабходна паслядоўна дзяліць яго на аснову другой СЛ, пакуль дзель ад дзялення не стане роўным нулю. Лік у новай СЛ утвараецца з астаткаў паслядоўнага дзялення, прычым старшы разрад – астатак ад апошняга дзялення. Разлікі вядуцца ў зыходнай СЛ.

Напрыклад, $115_{10} = 1110011_2$:



1.3. Пераклад дробаў

Для перакладу правільнага дробу з адной СЛ у другую выкарыстоўваецца метад паслядоўнага памнажэння: лічба пасля коскі паслядоўна памнажаецца на аснову другой сістэмы. Новы лік будзе складаць паслядоўнасць цэлых частак лічбаў, атрыманых памнажэннем. Прычым старшым разрадам будзе першая цэлая лічба. Колькасць такіх памнажэнняў будзе вызначацца патрэбнай дакладнасцю (колькасцю патрэбных пасля коскі разрадаў).

Прыклад. Перакласці лік з дзесятковай сістэмы ў двайковую з дакладнасцю 3 разрады ($0{,}385_{10} = 0{,}011_{2}$):

0,385
2
0,770
2
1,540
2
1,080

Калі ў старшым разрадзе атрымоўваецца 1, то яна ўмоўна выпадае і ў далейшым памнажэнні не ўдзельнічае. Пры перакладзе няправільнага дробу

асобна выкарыстоўваюць вядомыя метады перакладу цэлай і дробавай частак.

1.4. Формы ўяўлення лікаў у ЭЛВМ

Існуе дзве формы ўяўлення лікаў у любой пазіцыйнай СЛ: нармальная і натуральная. Нармальная – форма з плаваючай коскай (кропкай). Натуральная – форма з фіксаванай коскай (кропкай).

Прыклады прыведзены ў таблшцы 1.

Табліца 1

Сістэма лічэння	Форма запісу	
	Натуральная	Нармальная
Дзесятковая	115,275	$0{,}115275*10^{3}$
Двайковая	1110011,01	$0{,}11100110 1*10^{111}$

Разгледзім нармальную форму запісу: $\underbrace{0{,}115275}_{\text{мантыса}} * \underbrace{10}_{\text{аснова СЛ}}^{\overbrace{3}^{\text{парадак ліку}}}$

Зазначым, што мантыса мае нулявую цэлую частку, а аснова і парадак ліку запісваюцца ў адзінай СЛ.

2. КАДАВАННЕ ЛІКАЎ У ЭЛВМ

Існуе тры метада кадавання лікаў: прамы, дадатковы, адваротны. Пры арыфметычных аперацыях над двайковымі лікамі неабхода помятаць правілы множання і падсумоўвання апошніх: 1+0=0+1=1, 0+0=0, 1+1=10; 1*1=1, 0*1=1*0=0.

Прамы метад.

$[\mathbf{x}]_{\mathbf{пр}}$ – закадаваны лік **X** у прамым кодзе:

$$[\mathbf{x}]_{\mathbf{пр}} = \begin{cases} \mathbf{x}, & \mathbf{x} \geq 0, \\ 10^{\mathbf{j}} + |\mathbf{x}|, & \mathbf{x} < 0 \end{cases}$$

; **j** = 0 - для дробавага ліку ; **j** = n – 1 - для цэлага ліку .

Прыклад1 . Пераўтварыць у прамы код два дробавыя лікі ($\mathbf{x_1}$, $\mathbf{x_1}$):

$\mathbf{x_1}$ = 0,01011 , $\mathbf{x_2}$ = -0,01101.

Рашэнне:

Паколькі $\mathbf{x_1} > 0$, то $\mathbf{[x_1]_{пр}} = 0{,}01011$, між тым $\mathbf{x_2} < 0$, атрымоўваем $\mathbf{[x_2]_{пр}} = 10^0 + |-0{,}01101| = 1{,}01101$

Прыклад 2. Пераўтварыць у прамы код два цэлыя лікі, выкарыстоўваючы васьміразрадную сетку ЭЛВМ:

$\mathbf{x_1}$ = 1011, $\mathbf{x_2}$ = -1101.

Рашэнне:

$\mathbf{x_1} > 0$, тады $\mathbf{[x_1]_{пр}}$ = 00001011, а $\mathbf{x_2} < 0$ і $\mathbf{[x_2]_{пр}} = 10^{8-1} + |-1101| = 10000000 + |-1101| = 10001101$.

Дадатковы метад

$$\mathbf{[x]_{дад}} = \begin{cases} \mathbf{x}, & \mathbf{x} \geq 0 \\ \mathbf{x_{гр} - |x|}, & \mathbf{x} < 0 \end{cases}; \qquad \mathbf{x_{гр} = |x_{мах}| + 1} \text{ ці } \mathbf{x_{гр} = |x_{мах}| + 10^{-f}},$$

дзе $\mathbf{x_{гр}}$ – наперад зададзеная гранічная для вылічэнняў лічба, $\mathbf{|x_{мах}|}$ – максімальная для вылічэнняў лічба.

Прыклад 3. Пераўтварыць у дадатковы код два лікі: $\mathbf{x_1}$ = 85, $\mathbf{x_2}$ = -36 і знайсці іх суму ў дадатковым кодзе пры выкарыстанні двухраздраднай сеткі.

Рашэнне.

З прычыны таго, што мы выкарыстоўваем двухразрадную сетку ЭЛВМ, то прымаем $\mathbf{x_{мах}}$=99, а $\mathbf{x_{гр}}$ =99+1=100; $\mathbf{x_1} > 0$ і $\mathbf{[x_1]_{дад}}$=85, але $\mathbf{x_2} < 0$, таму $\mathbf{[x_2]_{дад}}$= 100-36=64.

Знойдзем суму гэтых лікаў у дадатковым кодзе: $\mathbf{[x_1]_{дад}}+\mathbf{[x_2]_{дад}}$=85+64=149 = 49. У атрыманым ліку старшым з'яўляецца трэці разрад. Але машына - двухразрадная, таму ён адкідаецца. Вынік – 49.

Прыклад 4. Пераўтварыць у дадатковы код два шасціразрадныя двайковыя лікі, выкарыстоўваючы васьміразрадню сетку ЭЛВМ:

$\mathbf{x_1}$ = 111011, $\mathbf{x_2}$ = -111101.

Рашэнне: $\mathbf{x_{1\ зых}} = 00111011$, $\mathbf{x_{2\ зых}} = -00111101$; $\mathbf{x_{мах}}=11111111$, а $\mathbf{x_{гр}} = =11111111+1=100000000$. Паколькі $\mathbf{x_1} > 0$, тады $\mathbf{[x_1]_{дад}} = 00111011$; $\mathbf{x_2} < 0$ і $\mathbf{[x_2]_{дад}} = =100000000 - 00111101 = 11000011$.

Для двайковай СЛ дадатковым кодам дробавага адмоўнага ліку з'яўляецца лік, у старшым разрадзе якога запісваецца 1 (прынцып адмоўнасці), усе астатнія разрады інвертуюцца і да малодшага дадаецца 1.

Прыклад 5. Пераўтварыць ў дадатковы код два дробавыя лікі: $\mathbf{x_1} = = 0{,}01011$ і $\mathbf{x_2} = -0{,}01101$.

Рашэнне: $\mathbf{x_1} > 0$, $\mathbf{[x_1]_{дад}} = 0{,}01011$; $\mathbf{x_2} < 0$, $\mathbf{[x_2]_{дад}} = 1{,}10010+0{,}00001=1{,}10011$.

Адваротны метад

$$\mathbf{[x]}_{\mathbf{адв}} = \begin{cases} \mathbf{x}, & \mathbf{x} \geq 0 \ ; \\ \mathbf{x} + \mathbf{x_{мах}}, & \mathbf{x} < 0 \ . \end{cases}$$

Прыклад 6. Пераўтварыць ў адваротны код два шасціразрадныя двайковыя лікі, з выкарыстаннем васьміразраднай сеткі ЭЛВМ: $\mathbf{x_1} = 101011$, $\mathbf{x_2} = -101101$.

Рашэнне: $\mathbf{x_{1\ зых}} = 00101011$, $\mathbf{x_{2\ зых}} = -00101101$, $x_{мах}=11111111$; $\mathbf{x_1} > 0$, тады $\mathbf{[x_1]_{адв}} = 00101011$, $\mathbf{x_2} < 0$ і $\mathbf{[x_2]_{адв}} = -00101101 +11111111 = 110100110$.

У заключэнні падкрэслім, што пры кадаванні дробавых адмоўных двайковых лікаў у старшым разрадзе (пры выкарыстанні любога метаду) пішацца 1.

3. ЛАГІЧНЫЯ АСНОВЫ ЭЛВМ

3.1.Асноўныя паняцці і лагічныя функцыі

Асноўнае паняцце **булевай алгебры** – *выказванне*.

Выказванне – гэта любое сцвярджэнне, якое можа быць або сапраўдным (true; 1), або памылковым (false; 0). Выказванне, значэнне якога не залежыць ад сувязяў і сапраўднасці іншых выказванняў называецца *простым*, іншае – *складаным*.

Для фармалізацыі (або матэматызацыі) лагічных выказванняў выкарыстоўваюцца *функцыянальна поўныя наборы* (ФПН) элементарных лагічных сувязяў. Разгледзім булевы функцыі *асноўнага функцыянальна поўнага набору* (АФПН).

Лагічнае "НЕ" (інверсія). Значэнне выхаднога параметра **Y** будзе сапраўдным толькі тады, калі значэнне ўваходнага параметра

X= false:**Y=** $\overline{\mathbf{X}}$ (чытаецца так: **Y** ёсць **не X**).

Уласцівасці лагічнага "НЕ": 1) няцотнае адмаўленне ёсць проста адмаўленне; 2) цотнае адмаўленне аргумента ёсць проста аргумент; 3) калі некалькі аргументаў роўныя паміж сабой, то і іх інверсіі будуць таксама роўныя; 4) інверсія абодзвух частак лагічнай роўнасці не змяняе гэту роўнасць.

Лагічнае "І" (кан'юнкцыя, лагічнае множанне). Кан'юнкцыяй **n** простых аргументаў (**X**) з'яўляецца такая складаная функцыя, якая прымае значэнне "true", калі значэнні ўсіх **n** аргументаў адпавядаюць "true". Пры **n**=2 функцыя запісваецца наступным чынам: $\mathbf{Y= X_1 \& X_2}$ або $\mathbf{Y= X_1 * X_2}$.

Уласцівасці лагічнага "І": 1) лагічнае множанне аднаго і таго ж аргумента самага на сябе дае значэнне гэтага аргумента; 2) лагічнае множанне любога аргумента на 0 дае 0; 3) лагічнае множанне любога аргумента на 1 дае 1 (true); 4) лагічнае множанне аргумента $\overline{\text{ }}$на яго інверсію дае 0 (false).

Лагічнае "ЦІ" ("АБО") (дыз'юнкцыя, лагічнае складанне). Дыз'юнкцыяй **n** простых аргументаў з'яўляецца такая складаная функцыя, якая прымае памылковае значэнне толькі тады, калі ўсе **n** аргументаў маюць значэнні "false": $\mathbf{Y= X_1+X_2+\ldots+X_n}$.

Уласцівасці лагічнага "ЦІ": 1) дыз'юнкцыя аднаго і таго ж аргумента роўная значэнню гэтага аргумента; 2) дыз'юнкцыя двух аргументаў, адзін з якіх памылковы (false), роўная другому аргументу; 3) дыз'юнкцыя двух аргументаў, адзін з якіх сапраўдны, будзе заўсёды сапраўднай (роўнай 1);

4) дыз'юнкцыя аргумента і яго адмаўлення заўсёды мае значэнне "true" (сапраўдная, роўная 1).

3.2.Асноўныя законы булевай алгебры

Сукупнасць аргументаў і знакаў, якія ўстанаўліваюць сувязі паміж аргументамі, называецца **лагічным выразам**. Асноўныя законы булевай алгебры рэгламентуюць парадак выканання лагічных аперацый. Сярод адзначаных законаў наступныя: 1) **перамяшчальны** - дыз'юнкцыю (кан'юнкцыю) некалькіх аргументаў можна запісаць пры любым распалажэнні (парадку) аргументаў:

$$Y = x_1+x_2 = x_2+ x_1;$$

$$Y = x_1*x_2 = x_2*x_1;$$

2) **спалучальны** - дыз'юнкцыю (кан'юнкцыю) **n** агументаў можна прадставіць у выглядзе сумаў (здабыткаў) гэтых аргументаў, частку якіх можна аб'яднаць:

$$Y = x_1+x_2+x_3= (x_1+x_2)+ x_3= x_1+(x_2+x_3)=\ldots;$$

$$Y = x_1*x_2*x_3= (x_1*x_2)* x_3= x_1*(x_2*x_3)=\ldots;$$

3) **размеркавальны** закон мае два варыянты (роды). *Першага роду*: кан'юнкцыяй дыз'юнкцыі некалькіх аргументаў на асобны аргумент з'яўляецца дыз'юнкцыя кан'юнкцый гэтага элемента на кожны з элементаў дыз'юнкцыі:

$$Y = (x_1+x_2)*x_3 = x_1* x_3 + x_2 * x_3;$$

другога роду: дыз'юнкцыяй асобнага аргумента на кан'юнкцыю іншых аргументаў з'яўляецца кан'юнкцыя (або некалькі) дыз'юнкцый гэтага асобнага элемента і іншых аргументаў, уваходзячых у зыходную кан'юнкцыю:

$$Y = x_1+x_2*x_3 = x_1+ x_3 * x_2 + x_3 ;$$

4) **інверсія (правіла дэ Моргана)** - інверсіяй кан'юнкцыі (дыз'юнкцыі) **n** аргументаў з'яўляецца дыз'юнкцыя (кан'юнкцыя) інверсій тых жа элементаў:

$$\overline{Y = x_1*x_2* \ldots *x_n} = \overline{x_1}+\overline{x_2}+ \ldots+ \overline{x_n},$$

$$\mathbf{Y = \overline{x_1 + x_2 + \ldots + x_n} = \bar{x}_1 * \bar{x}_2 * \ldots * \bar{x}_n,}$$

3.3. Вывады (правілы) з законаў булевай алгебры

1. *Правіла старшынства*.

Гэта правіла ўстанаўлівае парадак выканання аперацый над аргументамі лагічнага выразу. Калі ў лагічным выразе прадугледжана выкананне усіх аперацый (“НЕ”,”І”,”ЦІ”), то яны выконваюцца ў наступным парадку: 1)”НЕ”, 2)”І”, 3)”ЦІ”.

2. *Правілы склейвання*.

Азначэнне 1. **Элементарны здабытак** - гэта лагічны здабытак набору з **n** элементаў, якімі могуць з’яўляцца простыя аргументы або іх інверсіі. Напрыклад:

$\mathbf{F(x_1, x_2, x_3) = x_1 * x_2 * x_3}$, $\mathbf{F(x_1, x_2, x_3, x_4) = x_1 * x_2 * \overline{x_3 * x_4}}$, $\mathbf{F(x_1, x_2, x_3, x_4) = x_1 * x_2 * x_3 * x_4}$ – элементарныя здабыткі; але $\mathbf{F(x_1, x_2, x_3, x_4) = x_1 * \overline{x_2 * x_3} * x_3}$ – не з’яўляецца элементарным здабыткам.

Азначэнне 2. **Канстытуэнта адзінкі** - гэта элементарны здабытак усіх элементаў набору. Напрыклад, для набору $\mathbf{F(x_1, x_2, x_3, x_4)}$ канстытуэнтамі адзінкі будуць з’яўляцца наступныя : $\mathbf{\overline{x_1} * x_2 * x_3 * x_4}$, $\mathbf{x_1 * \overline{x_2} * \overline{x_3} * \overline{x_4}}$ і інш. З **n** элементаў можна скласці 2^n канстытуэнтаў адзінкі, прычым пры любых значэннях аргументаў набору значэнне толькі адной канстытуэнты адзінкі будзе адпавядаць true.

Азначэнне 3. **Ранг канстытуэнты** - гэта колькасць аргументаў, складаючых правую частку лагічнага выразу. Для апошняга прыкладу ранг канстытуэнты роўны чатыром.

Азначэнне 4. Два элементарных здабыткі называюцца **суседнімі**, калі яны маюць аднолькавы ранг і адрозніваюцца знакам інверсіі аднаго элемента.

Правіла склейвання для элементарных кан’юнкцый аднолькавага рангу: суму дзвух суседніх кан’юнкцый рангу **r** можна замяніць элементарным здабыткам рангу **r-1** агульных для зыходных кан’юнкцый элементаў. Напрыклад,

$$F(x_1, x_2, x_3, x_4)= x_1*x_2* x_3* x_4 + x_1*x_2* x_3* \bar{x}_4 = x_1*x_2* x_3 .$$

З улікам правілаў сіметрыі для булевай алгебры аналагічныя азначэнні можна сфармуляваць для дыз'юнкцый.

Азначэнне 5. **Элементарная сума** - гэта лагічная сума набору з **n** элементаў, якімі могуць з'яўляцца простыя аргументы або іх інверсіі.

Азначэнне 6. **Канстытуэнта нуля** - гэта элементарная сума ўсіх элементаў набору. Пры любых значэннях аргументаў набору значэнне толькі адной канстытуэнты нуля будзе адпавядаць false.

Азначэнне 7. Дзве элементарныя сумы называюцца **суседнімі**, калі яны маюць аднолькавы ранг і адрозніваюцца знакам інверсіі аднаго элемента.

Правіла склейвання для элементарных д'ызюнкцый аднолькавага рангу: здабытак дзвух суседніх дыз'юнкцый рангу **r** можна замяніць элементарнай сумай рангу **r-1,** агульных для зыходных дыз'юнкцый элементаў. Напрыклад:

$$F(x_1, x_2, x_3, x_4)=(x_1+x_2+x_3+ x_4)* (x_1+x_2+ x_3+ \bar{x}_4) = x_1+x_2+ x_3 .$$

3. Правілы паглынання.

Правіла паглынання для кан'юнкцый: лагічную суму дзвух элементарных кан'юнкцый розных рангаў, адна з якіх з'яўляецца састаўной часткай другой, можна замяніць кан'юнкцыяй меншага рангу. Напрыклад:

$$F(x_1, x_2, x_3, x_4)= x_1*x_2* x_3* x_4 + x_3* x_4 = x_3*x_4 .$$

Правіла паглынання для дыз'юнкцый. Лагічны здабытак дзвух элементарных дыз'юнкцый розных рангаў, адна з якіх з'яўляецца састаўной часткай другой, можна замяніць дыз'юнкцыяй меншага рангу. Напрыклад:

$$F(x_1, x_2, x_3, x_4)=(x_1+x_2+x_3+ x_4)* (x_1+ x_3+ x_4) = x_1+x_2+ x_4 .$$

4. Правілы разгортвання.

Правілы разгортвання выкарыстоўваюць, калі зыходны выраз трэба разгарнуць у суму канстытуэнт адзінкі або ў здабытак канстытуэнт нуля.

Разгортванне элементарных кан’юнкцый (як і дыз’юнкцый) ажыццяўляецца ў тры этапы:

1) элементы набору, якія адсутнічаюць у зыходным выразе, замяняюцца на лагічныя адзінкі;

2) кожная адзінка замяняецца на дыз’юнкцыю прамога і інверснага аргументаў;

3) выкарыстоўваецца размеркавальны закон першага роду для атрымання дыз’юнкцыі канстытуэнт адзінкі.

Разгортванне элементарных дыз’юнкцый:

1) элементы набору, якія адсутнічаюць у зыходным выразе, замяняюцца на лагічныя нулі;

2) кожны нуль замяняецца на кан’юнкцыю прамога і інверснага эламентаў;

3) выкарыстоўваецца размеркавальны закон другога роду для атрымання кан’юнкцыі канстытуэнт нуля.

4. АСНОВЫ СІНТЭЗУ КАМБІНАЦЫЙНЫХ СХЕМ

4.1. Этапы сінтэзу

У тэорыі лічбавых аўтаматаў (ЛА) існуе паняцце **лагічнага праектавання -** комплексу мерапрыемстваў па распрацоўцы лічбавага аўтамата (ЛА).

Сінтэз – адзін з этапаў праектавання; прадугледжвае стварэнне устройстваў на аснове функцый, апісваючых іх работу (лагічных сувязяў паміж уваходнымі і выхаднымі значэннямі ЛА).

Этапы праектавання:

1) апісанне функцыянавання аўтамата;
2) таблічнае апісанне сувязяў паміж уваходам і выхадам;
3) фармалізацыя таблічна зададзеных сувязяў (указанне лагічных сувязяў у выглядзе формулы);

4) аналіз формулаў з пункту гледжання ўсталяванай мінімальнай колькасці сувязяў паміж уваходам і выхадам;
5) сінтэз устройства: рэалізацыя яго ў выглядзе радыёэлементаў з функцыянальна поўнага набору.

Прыклад. Патрабуецца сінтэзаваць устройства, якое здзейснівае складанне двух двайковых лікаў (x_1, x_2) і выхаднымі сігналамі якога з'яўляецца разрад сумы (S) і разрад пераносу (P).

Па сутнасці сфармуляваная задача адпавядае заданню першага этапа.

Другі этап. Складзём табліцу сапраўднасці.

x_1	0	0	1	1
x_2	0	1	0	1
P	0	0	0	1
S	0	1	1	0

Трэці этап. Замена канкрэтных значэнняў выхадных сігналаў матэматычнымі сімваламі, як паказана ніжэй.

x_1	0	0	1	1
x_2	0	1	0	1
$f(x_1, x_2)$	$f_0(x_1, x_2)$	$f_1(x_1, x_2)$	$f_2(x_1, x_2)$	$f_3(x_1, x_2)$

Індэксы ў функцыях – двайковае ўяўленне нумара слупка ўваходных набораў. Замяняючы нулі на інверсныя значэнні аргумента, ў агульным выглядзе атрымаем:

$$f(x_1, x_2) = \overline{x_1} \cdot \overline{x_2} \cdot f_0 + \overline{x_1} \cdot x_2 \cdot f_1 + x_1 \cdot \overline{x_2} \cdot f_2 + x_1 \cdot x_2 \cdot f_3.$$

Выраз уяўляе дыз'юнкцыю канстатуэнты адзінкі. У такіх выразах заўжды ёсць складаемыя , тоесна раўныя 0 і 1, таму апошнюю формулу можна перапісаць у наступным выглядзе:

$f(x_1, x_2) = \bigvee_{j=1} \bigwedge_{i=1}^{n} x_i^{(1)}$ – **дасканалая дыз'юнктыўная нармальная форма** (ДДНФ) . Гэткае кананічнае раўнанне – форма запісу лагічных сувязяў паміж уваходам і выхадам. Для нашага прыкладу ДДНФ прыме выгляд

$$P(x_1, x_2) = x_1 \cdot x_2,$$

$$S(x_1, x_2) = \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}.$$

Агульнае правіла: *для запісу ДДНФ неабходна запісаць дыз'юнкцыю элементарных кан'юнкцый, якія адпавядаюць адзінкавым значэнням выхаднога сігналу, прычым аргументамі элементарных кан'юнкцый будуць прамыя значэнні, калі зыходны параметр роўны* ***1****, ці інверсныя значэнні, калі адпаведны аргумент роўны* ***0****.*

З улікам уласцівасці сіметрыі форма запісу ў выглядзе кан'юнкцыі элементарных дыз'юнкцый называюць **даскананай кан'юнктыўнай нармальнай формай (ДКНФ):** $f(x_1, x_2) = \bigwedge_{j=0} \bigvee_{i=1}^{n} x_i^{(0)}$ – кананічная форма запісу ДКНФ. Адпаведна ў нашым выпадку

$$P(x_1, x_2) = (x_1 + x_2) \cdot (x_1 + \overline{x_2}) \cdot (\overline{x_1} + x_2),$$

$$S(x_1, x_2) = (x_1 + x_2) \cdot (\overline{x_1} + \overline{x_2}).$$

Такім чынам, *для запісу лагічнай функцыі ў ДКНФ неабходна скласці кан'юнкцыю элементарных дыз'юнкцый, якія адпавядаюць нулявым значэнням выхадных сігналаў, прычым значэнні аргументаў, якія ўваходзяць у формулу, ўсталёўваюцца так, што нулявому значэнню зыходнага аргумента адпавядае адзінка (прамое значэнне ў формуле), а адзінкаваму – інверснае.*

4.2. Метады мінімізацыі лагічных функцый

Прааналізуем чацвёрты этап сінтэзу. Існуе тры асноўныя метады мінімізацыі: 1) разліковы; 2) разлікова-таблічны; 3) таблічны.

Рэалізацыя метадаў здзяйсняецца ў тры агульных этапы. Зыходным для мінімізацыі з'яўляецца запіс лагічнай функцыі ў ДДНФ або ў ДКНФ. На падставе выкарыстання аперацый склейвання прывядзём дасканалыя формы да **скарочаных**: ДДНФ - да **скДНФ**, ДКНФ – да **скКНФ**. Адна і тая ж элементарная кан'юнкцыя ці дыз'юнкцыя можа ўдзельнічаць у некалькіх працэдурах склейвання. Склейванне ажыццяўляецца да таго часу, пакуль не застануцца толькі **ізаляваныя элементы** (да якіх не прымяняюцца аперацыі склейвання). Элементарныя кан'юнкцыі ў скарочанай форме скДНФ называюцца **імплікантамі**, элементарныя дыз'юнкцыі ў скКНФ – **імпліцэнтамі.** На другім этапе адбываецца пераўтварэнне скарочаных форм у тупіковыя (скДНФ – тДНФ, скКНФ - тКНФ) . **Тупіковая форма** – такая форма запісу, якая адрозніваецца ад скарочанай адсутнасцю імплікант (імпліцэнт), не ўплываючых на значэнне сапраўднасці лагічных функцый. Апошні этап умоўна назавём **фактарызацыяй**. Фактарызацыя – змяненне тупіковай формы запісу, якая забяспечвае меншыя апаратныя затраты на рэалізацыю устройства.

4.2.1. Разліковы метад мінімізацыі

Мінімізацыя лагічных функцый у ДДНФ.

Прыклад. Мінімізаваць функцыю, зададзеную ў ДДНФ:

$$f_{\text{ДДНФ}} = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3 .$$

Атрымаем скДНФ (склейваюцца папарна кан'юнкцыі, якія падкрэслены аднолькавымі лініямі:першая і трэцяя, першая і чацвёртая і г.д.):

$$f = \underbrace{\overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot \overline{x_3}}_{\sim} + \underbrace{\overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3}_{\sim} = \underbrace{(\overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3)}_{\sim} + (\overline{x_1} \cdot x_2 \cdot \overline{x_3} +$$

$$+ x_1 \cdot x_2 \cdot x_3) + (\overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3) = (\overline{x_1} \cdot x_3 + \overline{x_1} \cdot x_3) \cdot (\overline{x_2} + x_2) + (\overline{x_1} \cdot x_2 + \overline{x_1} \cdot x_2) \cdot (\overline{x_3} +$$

$$+ x_3) + (\overline{x_2} \cdot x_3 + \overline{x_2} \cdot x_3) \cdot (\overline{x_1} + x_1) = \overline{x_1} \cdot x_3 + \overline{x_1} \cdot x_2 + \overline{x_2} \cdot x_3$$

Такім чынам, скДНФ прыме выгляд

$$f_{\text{скДНФ}} = \overline{x_1} \cdot x_3 + \overline{x_1} \cdot x_2 + \overline{x_2} \cdot x_3 .$$

Простая імпліканта (імпліцэнта) прымае значэнне аргументаў , якія ў яе ўваходзяць, калі элементарны здабытак роўны адзінцы і элементарная сума роўная нулю адпаведна.

Знойдзем тупіковую форму, для чаго паслядоўна будзем прымаць імпліканты роўнымі адзінцы і пры гэтым аналізаваць вынік. Няхай першая імпліканта роўная 1; гэта адбываецца, калі $x_1 = 0$ і $x_3 = 1$. Пры гэтым сума астатніх членаў будзе адпавядаць наступнаму:

$$\overline{x_1} \cdot x_2 + \overline{x_2} \cdot x_3 = 1 \cdot x_2 + \overline{x_2} \cdot 1 = x_2 + \overline{x_2} = 1.$$

Значэнне сапраўднасці першай імпліканты роўна адзінцы, пры гэтым значэнне астатняй сумы тасама адзінка, г.зн. першая імпліканта не ўплывае на вынік выразу. Тое ж самае робім адносна астаніх членаў: $\overline{x_1} \cdot x_2 = 1$ пры $x_1 = 0$ і $x_2 = 1$, тады

$$\overline{x_1} \cdot x_3 + \overline{x_2} \cdot x_3 = 1 \cdot x_3 + 0 \cdot x_3 = x_3,$$

$\overline{x_2} \cdot x_3 = 1$ пры $x_2 = 0$ і $x_3 = 1$, тады

$$\overline{x_1} \cdot x_3 + \overline{x_1} \cdot x_2 = \overline{x_1} \cdot 1 + \overline{x_1} \cdot 0 = \overline{x_1} .$$

Апошнія дзве імпліканты не з'яўляюцца лішкавымі і, па гэтай прычыне, увойдуць у тупіковую форму. Запішам яе:

$$f_{\text{тДНФ}} = \overline{x_1} \cdot x_2 + \overline{x_2} \cdot x_3 .$$

Пры пераўтварэнні скарочанай формы ў тупіковую могуць з'явіцца некалькі лішкавых элементаў. Адразу адкінуць іх нельга. Неабходна паэтапная праверка.

Мінімізацыя лагічных функцый у ДКНФ. Пры мінімізацыі лагічных функцый у ДКНФ першы і трэці этапы не адрозніваюцца ад разгледжанага (ДДНФ), але ёсць некаторыя асаблівасці для другога этапа: змяненне ў тупіковую форму здзяйсняецца выдаленнем лішкавых імпліцэнт, не ўплываючых на табліцу сапраўднасці (пры іх тоеснасці нулю, нулю павінны быць роўныя і кан'юнкцыі астатніх дыз'юнкцый).

Прыклад.

$$f_{\text{ДКНФ}} = (x_1 + x_2 + x_3)\cdot(\overline{x_1} + x_2 + x_3)\cdot(\overline{x_1} + \overline{x_2} + x_3)\cdot(\overline{x_1} + \overline{x_2} + \overline{x_3}).$$

Знойдзем скКНФ:

$$f = (x_1 + x_2 + x_3)\cdot(\overline{x_1} + x_2 + x_3)\cdot(\overline{x_1} + \overline{x_2} + x_3)\cdot(\overline{x_1} + \overline{x_2} + \overline{x_3}) = (x_1 + x_2 + x_3)\cdot(\overline{x_1} + x_2 + x_3)\times$$

$$\times(\overline{x_1} + x_2 + x_3)\cdot(\overline{x_1} + \overline{x_2} + x_3)\cdot(\overline{x_1} + \overline{x_2} + x_3)\cdot(\overline{x_1} + \overline{x_2} + \overline{x_3}) = (x_2 + x_3)\cdot(\overline{x_1} + x_3)\cdot(\overline{x_1} + \overline{x_2})$$

Атрымалі:

$$f_{\text{скКНФ}} = (x_2 + x_3)\cdot(\overline{x_1} + x_3)\cdot(\overline{x_1} + \overline{x_2}).$$

Для знаходжання тупіковай формы скарыстаем вышэйзгаданае правіла:

1) першая імпліцэнта $x_2 + x_3 = 0$, калі $x_2 = 0$ і $x_3 = 0$, пры гэтым

$$(\overline{x_1} + x_3)\cdot(\overline{x_1} + \overline{x_2}) = (\overline{x_1} + 0)\cdot(\overline{x_1} + 1) = \overline{x_1}.$$

Першая імпліцэнта ўплывае на вынік выразу, г.зн. не з'яўляецца лішкавай.

2) $\overline{x_1} + x_3 = 0$ пры $x_1 = 1$ і $x_3 = 0$ і,

$$(x_2 + x_3)\cdot(\overline{x_1} + \overline{x_2}) = (x_2 + 0)\cdot(0 + \overline{x_2}) = 0.$$

Такім чыным, другая імпліцэнта лішкавая.

3) $(\overline{x_1} + \overline{x_2}) = 0$ пры $x_1 = 1$ і $x_2 = 1$ і

$$(x_2 + x_3)\cdot(\overline{x_1} + x_3) = (1 + x_3)\cdot(0 + x_3) = x_3.$$

Апошняя імпліцэнта не з'яўляецца лішкавай. Запішам тКНФ:

$$f_{\text{тКНФ}} = (x_2 + x_3)\cdot(\overline{x_1} + \overline{x_2}).$$

4.2.2. Разлікова-таблічны метад мінімізацыі

Сутнасць метаду заключаецца ў тым, што пераход да тупіковай формы здзяйсняецца з дапамогай табліцы. Такія табліцы называюць **табліцамі Квайна** або **табліцамі пакрыццяў:** канстытуэнтна-імплікатная - пры утварэнні скДНФ і канстытуэнтна-імпліцэнтная - пры ўтварэнні скДНФ.

Прыклад. Функцыя зададзена ў ДДНФ:

$$f_{\text{ДДНФ}} = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3 .$$

Тады $f_{\text{скДНФ}} = \overline{x_1} \cdot x_3 + \overline{x_1} \cdot x_2 + \overline{x_2} \cdot x_3 .$

Табліца складаецца так, што яе слупкі ўтвараюць элементарныя кан'юнкцыі з дасканалай і скарочанай форм, як паказана ніжэй у табліцы 2.

Табліца 2

Імпліканты	Канстытуэнты			
	$\overline{x_1} \cdot \overline{x_2} \cdot x_3$	$\overline{x_1} \cdot x_2 \cdot \overline{x_3}$	$\overline{x_1} \cdot x_2 \cdot x_3$	$x_1 \cdot \overline{x_2} \cdot x_3$
$\overline{x_1} \cdot x_3$	+	-	+	-
$\overline{x_1} \cdot x_2$	+	-	-	+
$\overline{x_2} \cdot x_3$	-	+	+	-

Знак “+” азначае , што адпаведная імпліканта перакрывае адпаведную канстытуэнту. Як бачна з табліцы, апісаць лагічную функцыю можна дзвюма імплікантамі. Атрыманая функцыя будзе тупіковай:

$$f_{\text{тДНФ}} = \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 .$$

Для ДКНФ: $f_{\text{ДКНФ}} = (x_1 + x_2 + x_3) \cdot (\overline{x_1} + x_2 + x_3) \cdot (\overline{x_1} + \overline{x_2} + x_3) \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3})$;

$$f_{\text{скКНФ}} = (x_2 + x_3) \cdot (\overline{x_1} + x_3) \cdot (\overline{x_1} + \overline{x_2}) .$$

Імпліцэнтна-канстытуэнтная табліца прыме наступны выгляд (табл.3).

Табліца 3

Імпліцэнты	Канстытуэнты			
	$x_1 + x_2 + x_3$	$\overline{x_1} + x_2 + x_3$	$\overline{x_1} + \overline{x_2} + x_3$	$\overline{x_1} + \overline{x_2} + \overline{x_3}$

$x_2 + x_3$	+	+	-	-
$\overline{x_1} + x_3$	-	+	+	-
$\overline{x_1} + \overline{x_2}$	-	-	+	+

Такім чынам, для апісання лагічнай функцыі ў тупіковай форме дастаткова дзвюх імпліцэнт:

$$f_{\text{тКНФ}} = (x_2 + x_3) \cdot (\overline{x_1} + \overline{x_2}).$$

4.2.3. Таблічны метад мінімізацыі

Сутнасць метаду: пераўтварэнне ў тупіковую форму здзяйсняецца пры дапамозе універсальных табліц - **дыяграм Вейча-Карно**. Табліца складаецца з 2^n ячэек, дзе **n** - колькасць аргументаў функцыі. Імёны ячэек могуць быць адвольнымі, але кожная ячэйка (у некаторых выпадках – пара ячэек) у слупку або ў радку павінна адрознівацца ад суседняй толькі інверсіяй аднаго аргумента.

Вызначым памеры табліцы (вышыня-шырыня) у залежнасці ад **n**. Калі **n** - няцотны лік, табліца мае памеры $\mathbf{2^{(n-1)/2}}$ x $\mathbf{2^{(n+1)/2}}$, пры цотных **n** - $\mathbf{2^{n/2}}$ x $\mathbf{2^{n/2}}$. Напрыклад, **n** = 2; $f = f(x_1, x_2)$. Тады памеры табліцы 2×2:

	x_1	$\overline{x_1}$
x_2		
$\overline{x_2}$		

n=3; $f = f(x_1, x_2, x_3)$. Памеры табліцы 2×4:

	$\overline{x_2}$	$\overline{x_2}$	x_2	x_2
x_1				
$\overline{x_1}$				

$\overline{x_3}$ x_3 x_3 $\overline{x_3}$

З мэтай спрашчэння ў табліцу запісваюцца толькі адпаведныя значэнні функцыі: для дыз'юнктыўнай формы - адзінкі, а для кан'юнктыўнай - нулі з улікам сіметрыі лагічных выразаў у булевай алгебры. Прычым радкі і слупкі ячэек дыяграмы Вейча-Карно абазначаюцца не прамымі і інверснымі аргументамі, а адзінкамі і нулямі адпаведна.

Працэс мінімізацыі. $\mathbf{2^i}$ сумежных клетак, што складаюць прамавугольнік або квадрат, у якія ўнесены значэнні функцыі, можна замяняць на элемент меншага рангу, колькасць сімвалаў у якім на ***i*** адзінак менш. Калі гэты прамавугольнік (квадрат) складаюць элементы, якія адпавядаюць аднаму слупку ці радку, тады элементарную кан'юнкцыю (дыз'юнкцыю) можна паменшыць на адзін ранг: $r = n - 1$. Калі значэнні функцыі не складаюць прамавугольнік (квадрат), тады дасканалая форма будзе адпавядаць тупіковай: $r = n$. Калі ж згаданы вышэй квадрат складае ячэйкі, якія належаць да цэлай дыяграмы, элементы (канстытуэнты) дасканалай формы замяняюцца на адзін аргумент у прамым або інверсным кодзе.

Прыклад 1. Мінімізаваць функцыю, якая зададзена дыяграмай:

	x_1	$\overline{x_1}$
x_2	1	
$\overline{x_2}$	1	

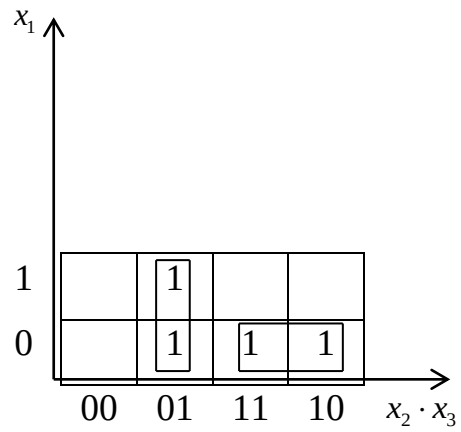
Атрымаем $x_1 \cdot x_2 + x_1 \cdot \overline{x_2} = x_1$.

На размяшчэнне значэнняў функцыі ў ячэйках уплывае адзін факт: значэнні, якія знаходзяцца на краях кожнага радка ці слупка, павінны адрознівацца інверсіяй аднаго элемента (па прынцыпу пераўтварэння дыяграмы ў гарызантальны ці вертыкальны цыліндр). Дзве сумежныя ячэйкі, размешчаныя па краях слупкоў і радкоў, лічацца сумежнымі, г.зн. функцыі ў гэтых ячэйках замяняюць на элементарныя функцыі меншага рангу.

Прыклад 2. Зададзена функцыя

$$f_{\text{ДДНФ}} = \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x_2} \cdot x_3$$

Складаем табліцу Вейча-Карно:



Пасля спрашчэнняў (паказана звычайным чынам: адзначаныя вышэй прамавугольнікі відавочны) атрымалі наступны выгляд функцыі, які адпавядае тупіковаму і супадае з атрыманым на падставе разліковага метаду:

$$f_{\text{тДНФ}} = \overline{x_1} \cdot x_2 + \overline{x_2} \cdot x_3 .$$

Аналіз дыяграм Вейча-Карно паказвае, што мінімальнай форме будзе адпавядаць мінімальная колькасць ячэек пры максімальным ліку ячэек, якія аб'ядноўваюцца. Адны і тыя ж ячэйкі могуць уваходзіць у некалькі прамавугольнікаў.

5.ВЫКАНАННЕ АРЫФМЕТЫЧНЫХ АПЕРАЦЫЙ У ЭЛВМ

5.1. Метады падсумоўвання двайковых лікаў

Пры складанні або адыманні (калі адно з складаемых адмоўнае) выконваецца аперацыя падсумоўвання, прычым сума фарміруецца з улікам знакавага разраду. Калі пры выкананні аперацыі падсумоўвання знакавая частка складаецца з двух разрадаў, то пры выкарыстанні дадатковага кода старшы разрад знакавай часткі адкідаецца. Пры выкарыстанні адваротнага кода адзінка старшага разраду знакавай часткі дадаецца да малодшага

разраду (цыклічны перанос). Знак ліку ў выніку выканання арыфметычнай аперацыі фарміруецца аўтаматычна (**0** - дадатны, **1** - адмоўны). Абсалютныя значэнні лікаў павінны быць такімі, каб не здарылася перапаўнення разраднай сеткі.

Прыклад 1. Падсумоўванне дадатнага і адмоўнага лікаў:

$X_1 = 0.10101; X_2 = -0.00101; X_3 = X_1 + X_2$.

Код	Прамы	Дадатковы	Адваротны
Прыклад вылічэнняў	$\begin{array}{r} {}_{-}0.10101 \\ 0.00101 \\ \hline 0.10000 \end{array}$	$\begin{array}{r} {}_{+}0.10101 \\ 1.11011 \\ \hline 10.10000 \end{array}$	1) $\begin{array}{r} {}_{+}0.10101 \\ 1.11010 \\ \hline 10.01111 \end{array}$ 2) $\begin{array}{r} {}_{+}0.01111 \\ 0.00001 \\ \hline 0.10000 \end{array}$
Вынік	$X_3 = 0.10000$	$X_3 = 0.10000$	$X_3 = 0.10000$

Прыклад 2. Падсумоўванне двух адмоўных лікаў:

$X_1 = -0.10101; X_2 = -0.00101; X_3 = X_1 + X_2$.

Код	Прамы	Дадатковы	Адваротны
Прыклад вылічэнняў	$\begin{array}{r} {}_{+}-0.10101 \\ -0.00101 \\ \hline -0.11010 \end{array}$	$\begin{array}{r} {}_{+}1.01011 \\ 1.11011 \\ \hline 11.00110 \end{array}$	1) $\begin{array}{r} {}_{+}1.01010 \\ 1.11010 \\ \hline 11.00100 \end{array}$ 2) $\begin{array}{r} {}_{+}1.00100 \\ 0.00001 \\ \hline 1.00101 \end{array}$
Вынік	$X_3 = -0.11010$	$X_3 = 1.00110$	$X_3 = 1.00101$

5.2. Спосабы множання лікаў з фіксаванай кропкай

Множанне дробавых і цэлых лікаў практычна нічым не адрозніваецца. Працэдура множання складаецца з двух асноўных этапаў: 1) вызначэнне знака здабытку; 2) вылічэнне самога здабытку ў выглядзе здабытку мідуляў аперандаў.

Вызначэнне знака здабытку выконваецца на аснове падсумоўвання знакавых разрадаў, прычым у атрыманай суме ўлічваецца толькі старшы разрад (1+1=0; 0+0=0; 1+0=0+1=1). Працэс множання ўяўляе сабой накапленне сумы - шматразовае складанне множымага, зрушанага адносна малодшага або старшага разраду аперанда. У адпаведнасці з гэтым існуюць 4

асноўныя способы множання: 1) малодшымі разрадамі наперад са зрухам накапляльных здабыткаў улева: 2) старшымі разрадамі наперад са зрухам накапляльных здабыткаў управа; 3) малодшымі разрадамі наперад са зрухам частковых сум управа; 4) старшымі разрадамі наперад са зрухам частковых сумаў улева.

Для множання лікаў не патрабуюцца спецыяльныя множнікі. Устройства множання складаецца з суматара, рэгістра зруху і устройства кіравання, якое ажыццяўляе сінхранізацыю суматара і рэгістра зруху. Разгледзім способы множання лікаў на прыкладах.

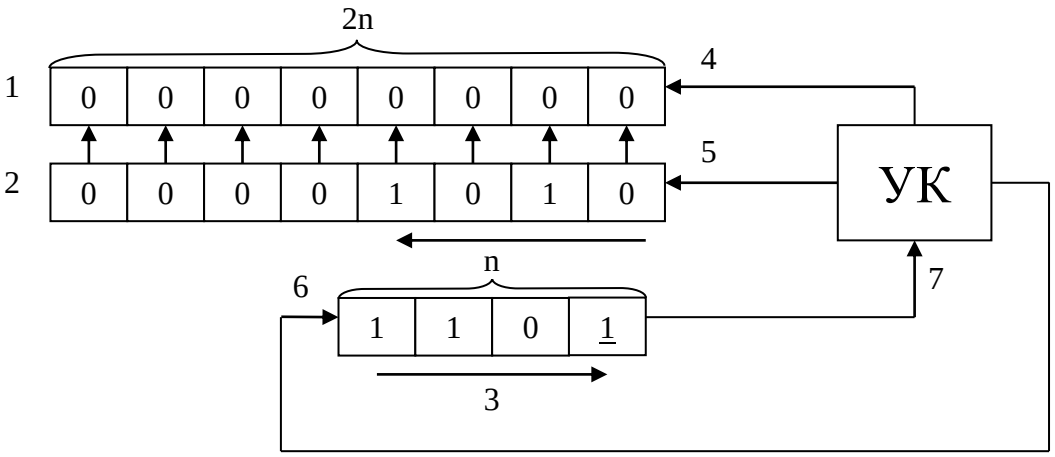
5.2.1. Множанне лікаў са зрухам частковых здабыткаў

Працэдура множання складаецца з некалікіх этапаў (табл. 4).

Табліца 4

Малодшымі разрадамі наперад	Старшымі разрадамі наперад
×1010 1101 1010 0000 1010 1010 10000010	×1010 1101 1010 1010 0000 1010 10000010

На кожным этапе выконваюцца аперацыя складання і аперацыя зруха. Разгледзім структурную схему множніка па способу малодшымі разрадамі наперад са зрухам частковых здабыткаў. На схеме лічбамі абазначаны: 1 – суматар, 2 – рэгістр зруху множымага, 3 – рэгістр зруху множніка, 4 – сігнал дазволу або забароны падсумоўвання, 5 – сігнал кіравання зрухам рэгістра множымага, 6 – сігнал кіравання зрухам рэгістра множніка, 7 – сігнал стану малодшага разраду рэгістра множніка; УК – устройства кіравання, **n** – колькасць разрадаў множымага і множніка.



Спачатку ў суматары 1 ва ўсіх разрадах - нулі, у рэгістр зруху 2 занесена множымае (малодшы разрад справа), у рэгістр 3 занесены множнік (малодшы разрад справа). Калі ў малодшым разрадзе рэгістра множніка знаходзіцца адзінка, то выконваецца паразраднае падсумоўванне двух лікаў з суматара 1 і рэгістра 2 і дабавляецца зрух рэгістраў 2 і 3 адпаведна ўлева і ўправа. Калі малодшы разрад рэгістра 3 нулявы, то выконваецца зрух без сумавання.

Прыклад. Разгледзім работу множніка на прыкладзе множання лікаў $X_1 = 1010$ і $X_2 = 1101$ (малодшы разрад падкрэслены; табл. 5).

Табліца 5

№ такта	Суматар 1	Рэгістр 2	Рэгістр 3	Аперацыі
1	00000000	00001010	110<u>1</u>	Падсумоўванне і зрух
2	00001010	00010100	011<u>0</u>	Зрух
3	00001010	00101000	001<u>1</u>	Падсумоўванне і зрух
4	00110010	01010000	000<u>1</u>	Падсумоўванне і зрух
5	10000010	10100000	0000	

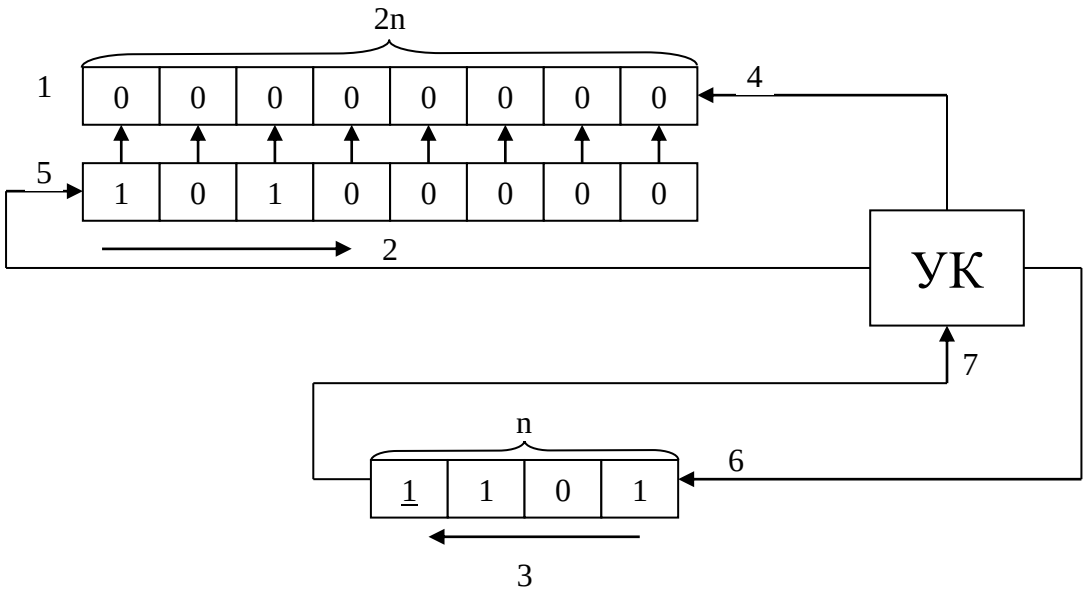
Пасля чатырох тактаў суматар будзе змяшчаць здабытак зыходных лікаў.

Аперацыі складання і зруху на кожным этапе могуць выконвацца паралельна. Аперацыя складання патрабуе больш часу, чым аперацыя зруху. Такім чынам, агуліны час, які патрабуецца для выканання аперацыі

множання, вызначаецца колькасцю разрадаў аперандаў і часам, неабходным для выканання аперацыі падсумоўвання.

Апаратныя затраты множніка па гэтаму спосабу складаюць **2n** разрадаў суматара, **2n** разрадаў рэгістра зруху множымага і **n** разрадаў рэгістра зруху множніка (без уліку устройства кіравання).

Структурная схема множніка па спосабу старшымі разрадамі наперад са зрухам частковых здабыткаў мае наступны выгляд, як паказаны ныжэй. З пункту гледжання функцыянальнай арганізацыі гэта схема аднолькавая з папярэдняй. Адрозніваюцца яны характарам сувязяў. Рэгістр зруху 2 ссоўвае множымае злева направа, рэгістр зруху 3 ссоўвае множнік зправа налева. Такім чынам, разрад множніка, які дазваляе або



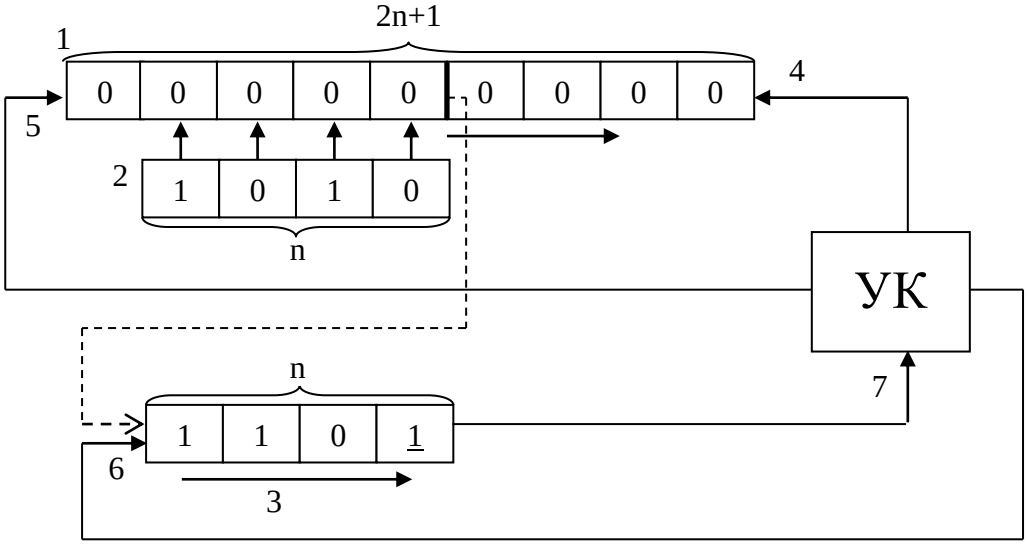
забараняе падсумоўванне, будзе крайнім левым. Частковыя здабыткі атрымліваюцца са зрухам управа. Паразраднае падсумоўванне на кожным з тактаў выконваецца па агульнах правілах.

5.2.2. Множанне лікаў са зрухам частковых сум

На кожным этапе множання выконваюцца аперацыя складання і аперацыя зруху частковай сумы. Пры рэалізацыі гэтага спосабу можа адбывацца перапаўненне разраднай сеткі ЭЛВМ.

Малодшымі разрадамі наперад	Старшымі разрадамі наперад
×1010 1101 +00000000 1010 →10100000 +01010000 0000 →01010000 +00101000 1010 →11001000 +01100100 1010 →100000100 10000010	×1010 1101 +00000000 1010 ←00001010 +00010100 1010 ←00011110 +00111100 0000 ←00111100 +01111000 1010 10000010

Разгледзім структурную схему множніка па спосабу малодшымі разрадамі наперад, са зрухам частковых сум. На схеме лічбамі абазначаны: 1



- суматар і рэгістр зруху частковай сумы, 2 - рэгістр множымага, 3 - рэгістр зруху множніка, 4 - сігнал дазволу або забароны падсумоўвання, 5 - сігнал кіравання зрухам частковай сумы, 6 - сігнал кіравання зрухам рэгістра

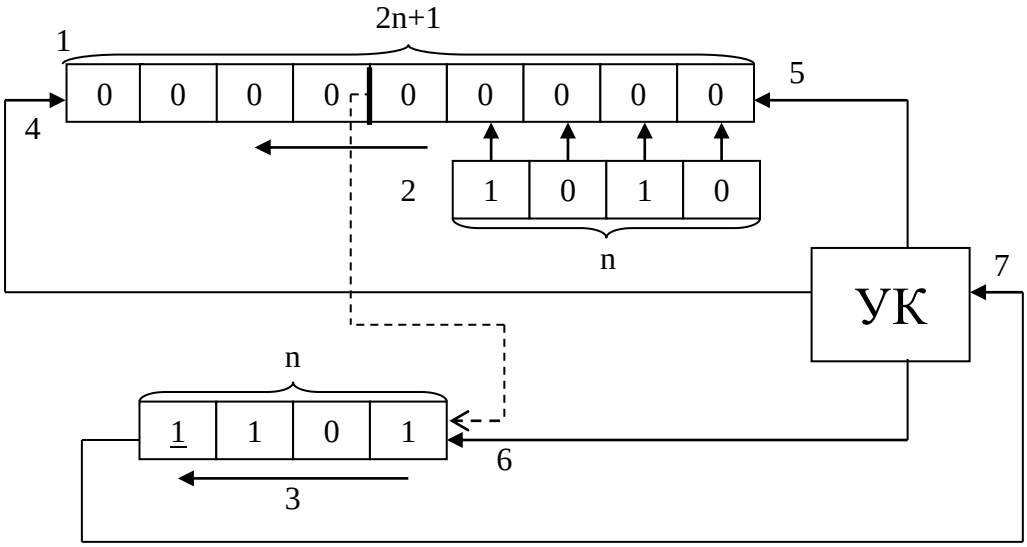
множніка, 7 - сігнал стану малодшага разраду рэгістра множніка, УК – устройства кіравання, **n** – колькасць разрадаў множымага і множніка.

Устройства кіравання, прызначэнне яго сігналаў, а таксама арганізацыя рэгістра 3 аднолькавыя з папярэднімі схемамі. Рэгістр 1, акрамя падсумоўвання выконвае зрух частковай сумы; з улікам магчымага перапаўнення ён дапоўнены адным разрадам. З кожным тактам рэгістры 1 і 3 ажыццяўляюць зрух змесціва ўлева на адзін разрад, таму мае сэнс малодшыя разрады рэгістра 1 заносіць замест старшых разрадаў рэгістра 3, а не ў правую палову рэгістра 1.

У адрозненне ад папярэдніх спосабаў, аперацыі падсумоўвання і зруху не могуць ажыццяўляцца паралельна. Час выканання адного такта – сума часу аперацыі падсумоўвання і часу аперацыі зруху. Такім чынам, множанне па спосабу са зрухам частковых сум займае больш часу, чым множанне папярэднімі спосабамі.

Апаратныя затраты множніка па гэтаму спосабу складаюць **n+1** разрадаў рэгістра зруху частковых сум, **n** разрадаў рэгістра зруху множніка і **n** разрадаў рэгістра множымага (без уліку устройства кіравання). У параўнанні з папярэднімі схемамі апаратныя затраты для множніка па спосабу са зрухам частковых сум меншыя.

Структурная схема множніка па спосабу старшымі разрадамі наперад са зрухам частковых сум мае наступны выгляд.



Па прынцыпу арганізацыі, апаратных затратах і хуткасці выканання множання схема аднолькавая з папярэдняй.

5.2.3. Множанне лікаў у дадатковых кодах

У большасці сучасных ЭЛВМ для множання лікаў выкарыстоўваецца дадатковы код. Пры гэтым для атрымання правільнага выніку неабходна ўлічваць папраўку. Знойдзем велічыню гэтай папраўкі для любога выразу выгляду

$$X_3 = X_1 \cdot X_2 .$$

Для пераводу адмоўнага ліку ў дадатковы код будзем карыстацца формулай

$$X^{\partial a\partial.} = C^j - |X|,$$

дзе C – аснова сістэмы лічэння, j – колькасць цэлых разрадаў. Паколькі знак здабытку вызначаецца палсумоўваннем па модулю 2 знакавых разрадаў, то будзем разглядаць множанне мадуляў аперандаў. Магчымы некалькі выпадкаў:

1) $X_1 > 0$, $X_2 > 0$,

$$X_3^{\partial a\partial.} = X_1 \cdot X_2 .$$

Папраўка да выніку $\Delta_1 = 0$.

2) $X_1 < 0$, $X_2 > 0$,

$$X_3^{\partial a\partial.} = \left(C^j - |X_1|\right) \cdot X_2 ,$$

$$X_3^{\partial a\partial.} = C^j \cdot X_2 - |X_1| \cdot X_2 .$$

Але сапраўдны вынік мае выгляд

$$X_3^{\partial a\partial.} = C^{2 \cdot j} - |X_1| \cdot X_2 .$$

Тады папраўка да выніку:

$$\Delta_2 = C^{2\cdot j} - C^j \cdot X_2 = C^j \cdot \left(C^j - X_2\right).$$

3) $X_1 > 0$, $X_2 < 0$,

$$X_3^{\partial a\partial.} = X_1 \cdot \left(C^j - |X_2|\right),$$

$$X_3^{\partial a\partial.} = C^j \cdot X_1 - X_1 \cdot |X_2|$$

і , аналагічна папярэдняму выпадку,

$$\Delta_3 = C^{2\cdot j} - C^j \cdot X_1 = C^j \cdot \left(C^j - X_1\right).$$

4) $X_1 < 0$, $X_2 < 0$,

$$X_3^{\partial a\partial.} = \left(C^j - |X_1|\right) \cdot \left(C^j - |X_2|\right),$$

$$X_3^{\partial a\partial.} = C^{2\cdot j} - C^j \cdot |X_2| - C^j \cdot |X_1| + |X_1| \cdot |X_2|,$$

$$X_3^{\partial a\partial.} = C^j \left(C^j - |X_2| - |X_1|\right) + |X_1| \cdot |X_2|.$$

Улічваючы агульны выгляд сапраўднага выніку, можна паказаць, што папраўка ў гэтым выпадку вылічаецца па формуле

$$\Delta_4 = C^j - |X_2| - |X_1|.$$

Такім чынам, здабытак двух лікаў у дадатковым кодзе будзе мець наступны агульны выгляд:

$$X_3^{\partial a\partial.} = X_1^{\partial a\partial.} \cdot X_2^{\partial a\partial.} + \Delta.$$

Прыклад. Разгледзім множанне двух лікаў у дадатковым кодзе. Знойдзем здабытак $X_1 = 1010$ і $X_2 = -0011$:

1) дадатковыя коды лікаў: $X_1^{\partial a\partial.} = 01010$, $X_2^{\partial a\partial.} = 11101$;

2) знак здабытку: $0 \oplus 1 = 1$;

3) здабытак мадуляў лікаў:

$$\begin{array}{r} \times 1010 \\ 1101 \\ \hline 1010 \\ 0000 \\ 1010 \\ 1010 \\ \hline 10000010 \end{array}$$

4) папраўка: $\Delta = 2^4 \cdot \left(2^4 - |X_1|\right) = 10000 \cdot (10000 - 1010) = 1100000$;

5) модуль здабытку: $\left|X_3^{\partial a\partial.}\right| = 10000010 + 1100000 = 11100010$;

6) вынік: $X_3^{\partial a\partial.} = 111100010$.

6. МЕТАДЫ ПАВЫШЭННЯ НАДЗЕЙНАСЦІ АПРАЦОЎКІ ІНФАРМАЦЫІ Ў ЭЛВМ

6.1. Фактары, паўплываючыя на надзейнасць ЭЦВМ. Асноўныя паняцці

Надзейнасць любога радыёэлекторннага устройства – гэта яго характарыстыка, якая адлюстроўвае магчымасць гэтага устройства выдаваць (на выхадзе) беспамылковыя даныя . Сярод паказчыкаў, якія характарызуюць надзейнасць, асноўнымі з'яўляюцца наступныя: *імавернасць безадмоўнай працы* – імавернасць таго, што ў межах зададзенай напрацоўкі аб'екту адмова не ўзнікне; *сярэдняя напрацоўка да адмовы* – матэматычнае чаканне напрацоўкі аб'екту да першай адмовы; *інтэнсіўнасць адмоў* - умоўная шчыльнасць імавернасці ўзнікнення адмовы аб'екта.

Надзейнасць залежыць ад мноства фактараў, асноўнымі з якіх з'яўляюцца якасць тэхналогіі стварэння устройстваў і ўмовы, ў якіх гэтыя устройствы выкарыстоўваюцца (электрамагнітнае становішча, памехі, тэмпература і г.д.).

Для сістэмаў і устройстваў вылічальнай тэхнікі адмоўнымі наступствамі ўздзеяння вышэйадзначаных фактараў з'яўляюцца *збоі (кароткачасовыя адмовы*) і *адмовы*, звязаныя з немагчымасцю дадзенага аб'екту выконваць свае функцыя . Вынікам збояў і адмоў з'яўляюцца інфармацыйныя памылкі : *арыфметычныя* – узнікаюць пры выкананні арыфметычных аперацый элементамі або блокамі, якія маюць адмовы, *памылкі перадачы і захоўвання двайковай інфармацыі.* Апошнія ўзнікаюць значна часцей, таму метады іх нейтралізацыі (выяўлення і карэкцыі) разгледзім больш падрабязна.

6.2. Метады карэкцыі інфармацыйных памылак

6.2.1. Сутнасць метадаў

Найбольш эфектыўнымі з’яўляюцца метады перашкодаўстойлівага кадавання інфармацыі. Пры гэтым асноўнае інфармацыйнае слова даўжынёй у *k* разрадаў дадаецца *r* разрадамі: $k + r = n$, дзе *n* – агульная даўжыня слова.

Прымяненне перашкодаўстойлівага кадавання заснавана на тым, што ўсе апрацоўваемыя словы, колькасць якіх $N_{б.п.}$(без памылак), складаюць *дазволеныя камбінацыі.* Пры з’яўленні памылак ствараюцца і камбінацыі, якія не ўваходзяць у лік дазволеных (забароненыя камбінацыі). *Памылка* – змяненне значэння аднаго ці некалькіх разрадаў на адваротныя. *Кратнасць памылак* – колькасць памылак. Напрыклад, калі 1101 – без памылак, то слова 1001 будзе з адной памылкай (кратнасць памылак – 1).

Прыкладам можа з'яўляцца змяненне кантрольнага разраду, які фарміруецца па ўмове цотнасці або няцотнасці адпаведных інфармацыйных разрадаў: калі колькасць адзінак інфармацыйнага слова няцотна, тады ў кантрольны разрад запісваецца 1 (дадатак да цотнасці), а калі ў інфармацыйным слове цотная колькасць адзінак - у кантрольны разрад заносіцца 0. *Прыклад:*

10101110 **1** – без памылак,
k разрадаў кантрольны разрад па модулю 2;

10001011 **0** – з памылкамі (у кантрольным разрадзе 0, а не 1, як у пачатковым слове).

Адзінкавы дадатковы разрад дазваляе выяўляць аднакратныя памылкі ці няцотную колькасць памылак.

6.2.2. Коды для выяўлення і выпраўлення памылак

Кодавае слова – паслядоўнасць сімвалаў, агульная колькасць якіх роўная *n*, складзеных з *k* інфармацыйных і *r* праверачных сімвалаў: k + r = n

Вагой кодавага слова **ω** з'яўляецца колькасць адзінкавых разрадаў у слове. Напрыклад, **ω**(1101) = 3. Мінімальная колькасць пазіцый, у якіх адрозніваюцца 2 кодавых словы, ёсць *кодавая адлегласць ці адлегласць Хэмінга* і абазначаецца звычайна літарай ***d***. Кодавая адлегласць паміж двума словамі вызначаецца падсумоўваннем па модулю 2 адпаведных разрадаў гэтых слоў. Напрыклад, знойдзем кодавую адлегласць паміж двума словамі: $x_1 = 10111$ і $x_2 = 11011$. Пасля падсумоўвання па модулю 2 адпаведных разрадаў дадзеных слоў, атрымаем: æ = 01100 ; **ω**(01100)=2 – кодавая адлегласць паміж x_1 і x_2.

Прынцып выяўлення памылак заключаецца ў існаванні адрозненняў паміж камбінацыямі з памылкамі (забароненыя камбінацыі) і без памылак (дазволеныя камбінацыі). Для выяўлення і памылак у кодавай паслядоўнасці мінімальная адлегласць паміж словамі павінна быць не менш чым ***i*** +1: $d_{\min} \geq i+1$. Неабходная ўмова - $d_{\min} = i+1$, дастатковая ўмова - $d_{\min} = i+2$.

Паколькі колькасць разрадаў, у якіх адрозніваюцца ўсе кодавыя камбінацыі паміж сабой, можа быць рознай, тады існуе мінімальна неабходная адлегласць, якая забяспечвае выяўленне і выпраўленне памылак. Для выпраўлення і памылак мінімальная кодавая адлегласць павінна быць не менш за $2 \cdot i + 1$. Такім чынам, пры i =1 $d_{\min} = 3$.

Прасцейшы код, які выяўляе і выпраўляе памылкі, называецца *кодам Хэмінга* (SEC-SED-code).

Код Хэмінга. Колькасць праверачных разрадаў кодавага слова вызначаецца па наступнай формуле: $r \geq \log_2 k + 1$.

З улікам таго, што $d_{\min} = 3$ і $k + r = n$, для розных k можна знайсці n і r. Асноўны прынцып пабудовы кода, на падставе якога правяраюцца разрады, заключаецца ў r кратным правядзенні згорткі па модулю 2 акрэсленай часткі інфармацыйных сімвалаў, г.зн. правяраемыя сімвалы павінны быць сфарміраваны так, каб па ім можна было адназначна ідэнтыфікаваць месца памылкі. Прынцып фарміравання базуецца на двайковым уяўленні n лікаў, як гэта прыведзена ў наступнай табліцы 6 для n =16.

Табліца 6

Першае паўслова	Другое паўслова	Трэцяе паўслова	Чацвёртае паўслова
0	**0**	**0**	**0**
0	**0**	**0**	**1**
0	**0**	**1**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
0	**1**	**0**	**1**
0	**1**	**1**	**0**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**0**	**0**	**1**
1	**0**	**1**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**0**	**1**
1	**1**	**1**	**0**
1	**1**	**1**	**1**

Фарміраванне паўслоў. Першае паўслова можа складаць любая палова зыходных даных слова. Другое – палова разрадаў, якія не ўвайшлі ў першае паўслова і палова, якая ўвайшла ў другое паўслова. І гэтак далей для наступных паўслоў. Згодна з табліцай, сімвалы, якія уваходзяць у адпаведнае паўслова, адзначаны адзінкамі, а разрады, якія не ўвайшлі – нулявымі лічбамі. Радкі матрыцы, што фарміруе паўсловы, у якіх маецца толькі па аднаму адзінкаваму сімвалу, мэтазгодна адвесці пад кантрольныя разрады.

Алгарытм вылічэння кантрольных разрадаў задаецца праверачнай матрыцай, якая абазначаецца наступным чынам: $H_{n,k} = [A; I]$, дзе $\boldsymbol{A}$ – матрыца, радкі якой маюць вагу больш за 1, $\boldsymbol{I}$ – матрыца з радкамі вагой 1. Тады матрыцу $\boldsymbol{H}$ можна перапісаць так:

$$H = \overbrace{\begin{matrix} 0&0&0&0&1&1&1&1&1&1&1 \\ 0&1&1&1&0&0&0&1&1&1&1 \\ 1&0&1&1&0&1&1&0&0&1&1 \\ 1&1&0&1&1&0&1&0&1&0&1 \end{matrix}}^{A} \quad \overbrace{\begin{matrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{matrix}}^{I}$$

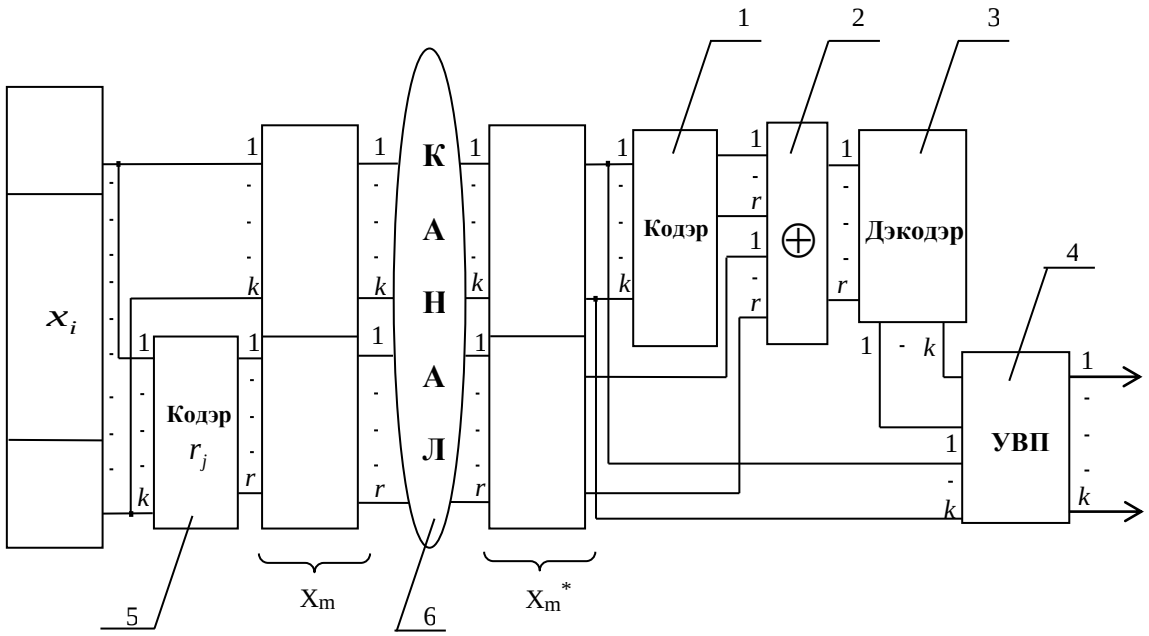
Асаблівасці: 1) матрыца I – матрыца з адзінкавай галоўнай дыяганаллю; 2) рэальна колькасць слупкоў у A-матрыцы $k = 11$, у I-матрыцы $r = 4$. Такім чынам, $n = 15$. Нулявы вектар-слупок (нулявы радок у табліцы

1) пры апрацоўцы кодавых слоў адпавядае выпадку адсутнасці памылак. Праверачны сімвал кодавага слова r_j фарміруецца так, што робіцца вылічэнне

$$r_j = \sum_{i=1}^{k} h_{ij} x_i \quad , \quad \text{дзе } j = \overline{1,r} \text{ і } h_j \text{ – вектар-слупок матрыцы } A.$$

На практыцы гэта ёсць згортка па модулю 2 разрадаў, адпавядаючых адзінкавым сімвалам у адпаведнным радку матрыцы A: r_1 роўны згортцы па модулю два сімвалаў з 5 па 11 разрад; r_2 роўны згортцы па модулю два сімвалаў з 2 па 4 і з 8 па 11 разрады і г.д.

Механізм выяўлення і выпраўлення памылак з дапамогай кода Хэмінга. Умоўна працэс апрацоўкі слова падзелім на два этапа: 1) этап кадавання слова і 2) этап дэка-давання слова (выяўлення і выпраўлення памылак). Працэс можна ўявіць на падставе наступнай структурнай схемы.



На мал. 1,5 – кодэры; 2 – фарміравальнік прыкметы памылкі; 3 – дэкодэр; 4 – устройства выпраўлення памылак; 6 – канал перадачы інфармацыі.

Кодэры 1 і 5 – абсалютна ідэнтычныя схемы. Яны могуць будавацца на суматарах па модулю 2. Колькасць двухуваходных (аднаразрадных) суматараў у кожным з кодэраў вызначаецца вагой **A**-матрыцы: $l_{\oplus} = w(A) - r$.

Для вылічэння аднаго з праверачных сімвалаў (напрыклад, першага з прыведзенай праверачнай матрыцы) неабходна згарнуць па модулю 2 сімвалы ад 5 па 11 уключна. Метад іх злучэння можа быць або ланцуговым, або пірамідальным.

6.3. Выяўленне і выпраўленне памылак у арыфметычных устройствах

За арыфметычныя будзем прымаць памылкі, узнікаючыя ў устройствах сумавання, множання, дзялення і адымання двайковых лікаў. Коды Хэмінга не могуць выявіць такія памылкі, таму што няма адназначнай сувязі паміж праверачнымі разрадамі аперандаў і праверачнымі разрадамі выніка арыфметычнай аперацыі. Метады кантролю памылак у арыфметычных устройствах заснаваны на аналізе згорткі па модулю аперандаў і выніка.

У кантрольныя разрады пры выкананні арыфметычных аперацый запісваецца астатак ад дзялення аперандаў і выніка на модуль. Асаблівасць у тым, што сума астаткаў аперандаў роуная астатку сумы аперандаў і здабытак астаткаў аперандаў роўны астатку здабытка. Напрыклад:

$$X_1 = 13,\ r_1 = X_1 \bmod 3 = 13 \bmod 3 = 1;$$

$$X_2 = 16,\ r_2 = X_2 \bmod 3 = 16 \bmod 3 = 1;$$

$$X_3 = X_1 + X_2 = 13 + 16 = 29,$$

$$r_3 = X_3 \bmod 3 = 29 \bmod 3 = 2 = r_1 + r_2.$$

$$X_3 = X_1 \cdot X_2 = 13 \cdot 16 = 208,$$

$$r_3 = X_3 \bmod 3 = 208 \bmod 3 = 1 = r_1 \cdot r_2.$$

Калі пры вылічэнні кантрольных разрадаў сумы або здабытка атрымліваецца лік большы за модуль, то і ён бярэцца па модулю.

Пры вылічэнні астаткаў да ўвагі можа прымацца не толькі лік (лікавы код), але і сума лічбаў (лічбавы код). Напрыклад:

$$18 \Rightarrow 1+8=9.$$

У электронных лічбавых вылічальных машынах астаткі ад дзялення лікаў на модуль знаходзяцца з дапамогай аперацый над лічбавымі кодамі. Значэнне модуля знаходзіцца па формуле

$$d_m = b^i - 1,$$

дзе d_m – модуль, b – аснова сістэмы лічэння, $i = 1,2,3...$ Каэфіцыент ***i*** уплывае на працэс вылічэння астаткаў, так як набор лічбаў дзеліцца на групы па ***i*** разрада.

Прыклад. Устанавіць правільнасць выканання арыфметычнай аперацыі, калі $X_1 = 1010$, $X_2 = 1101$, $X_3 = X_1 + X_2 = 10111$.

Знаходзім праверачныя разрады для аперандаў і выніка, карыстаючыся дзвухразраднай сеткай:

$$X_1 = 1010 = 10\_10,$$

$$r_1 = 10+10 = 100 = 01\_00 = 01+00 = 01;$$

$$X_2 = 1101 = 11\_01,$$

$$r_2 = 11+01 = 100 = 01\_00 = 01+00 = 01;$$

$$X_3 = 10111 = 01\_01\_11,$$

$$r_3 = 01+01+11 = 101 = 01\_01 = 01+01 = 10.$$

Параўноўваем суму праверачных разрадаў аперандаў з праверачнымі разрадамі сумы:

$$r_1 + r_2 = 01+01 = 10 = r_3.$$

Такім чынам, аперацыя складання лікаў X_1 і X_2 выканана правільна.

ЗМЕСТ

СПІС ЛІТАРАТУРЫ

1. Лысиков Б.Г. Арифметические и логические основы цифровых автоматов. – Мн: Вышэйшая школа. – 1980. – 336 с.
2. Урбанович П.П., Алексеев В.Ф., Верниковский Е.А. Избыточность в полупроводниковых интегральных микросхемах памяти. – Мн: Навука і тэхніка. – 1995. – 320 с.
3. Верниковский Е. А., Урбанович П.П. Статистические характеристики отказов запоминающих элементов в микросхемах памяти// Электронная техника. Сер. 3: Микроэлектроника. – 1989. – Вып. 1 (130). – С. 61-63.
4. Урбанович, П. П. Модель распределения дефектных запоминающих элементов на кристаллах БИС ЗУ// Изв. вузов. Радиоэлектроника. – 1986. – Т. 29. – № 9. – С. 92-95