* **Критерии выбора системы счисления.**

*(Любая система* счисления должна обеспечивать:

-возможность представления любого числа в рассматриваемом диапазоне величин;

-единственность этого представления;

-простоту оперирования числами.

Различают два типа систем счисления: *непозиционные и позиционные*.

*Основание(базис)-*r позиционной системы счисления - максимальное количество различных знаков или символов, используемых для изображения числа в данной системе счисления. Таким образом, основание может быть любым числом, кроме 1 и бесконечности.

*Длина числа* – количество позиций(разрядов) в записи числа. В технической реализации под длиной числа понимается длина разрядной сетки.

*Диапазон представления чисел* в заданной системе счисления – интервал числовой оси, заключенный между максимальным и минимальным числами, представленными при заданной длине разрядной сетки.)

Сформулируем требования, которым должна удовлетворять система счисления:

***Простота технической реализации***. Для хранения чисел в той или иной системе счисления используютсяn-позиционные запоминающие элементы. Элемент будет тем проще, чем меньше состояний требуется для запоминания цифры числа, то естьчем меньше основание системы счисления.

***Наибольшая помехоустойчивость кодирования цифр.*** Очевидно, что при наложении помехи на основной сигнал, соответствующий некоторой цифре, наиболее вероятна ошибка в устройствах, для которых используется система счисления с наибольшим основанием. Следовательно, при увеличении основания помеха может привести к искажению числа

***Минимум оборудования.***

***Простота арифметических действий***. Чем меньше цифр в системе счисления, тем проще арифметические действия над ними. Таблицы для выполнения четырех арифметических операций будут усложняться с увеличением основания системы счисления. Это можно принять за косвенное доказательство выдвинутого положения.

***Наибольшее быстродействие***.

Наиболее удобной системой счисления для работы человека является десятичная система счисления. Но внутри ЭВМ для выполнения арифметических операций числа из десятичной системы счисления требуется переводить во внутреннюю систему счисления. Для системы счисления с основанием, большим 10 появляются новые цифры. Таким образом, система счисления должна иметь минимальное число цифр, так как в этом случае можно пользоваться младшими цифрами десятичной системы счисления.

***Возможность представления любого числа в рассматриваемом диапазоне величин***.

***Единственность представления числа***.

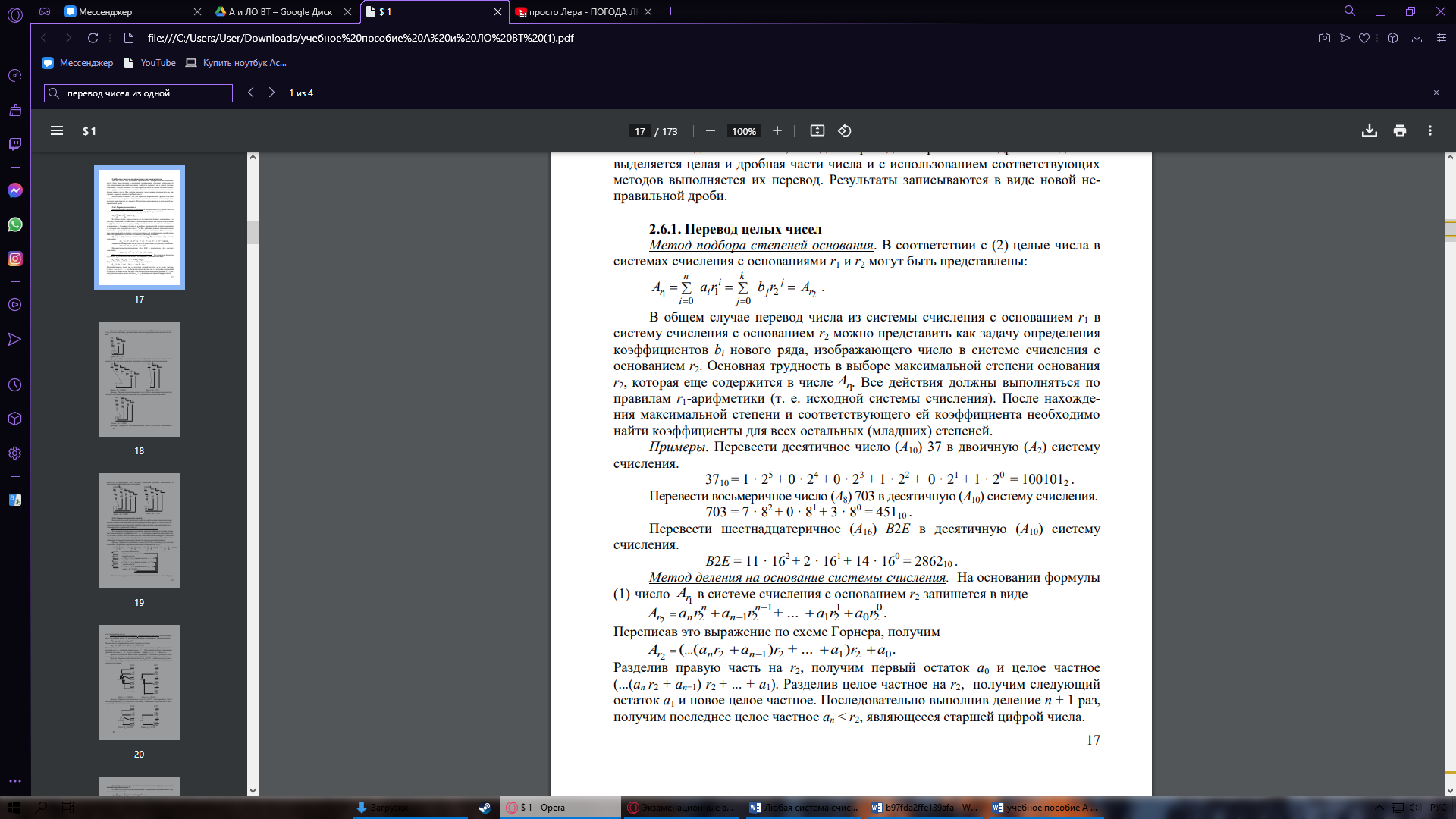
* **Перевод чисел из одной системы счисления в другую.**

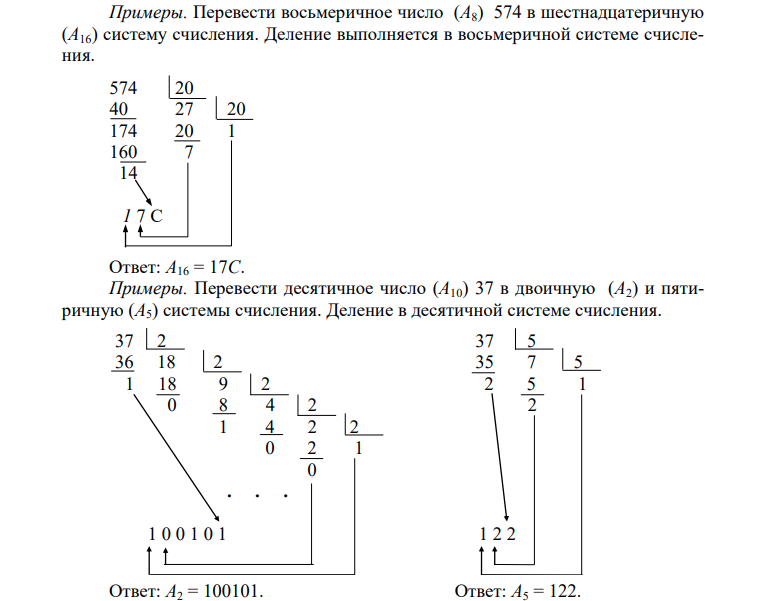
Так как числа, над которыми производятся арифметические операции, могут быть представлены в различных позиционных системах счисления, то для выполнения действий над ними требуется привести их к одной системе счисления. Следует помнить, что при переводе числа из одной системы счисления в другую количественное значение числа не изменяется, а меняется только форма записи числа. Все методы перевода чисел можно подразделить на две группы: перевода целых и дробных чисел.

***Перевод целых(дробных) чисел***

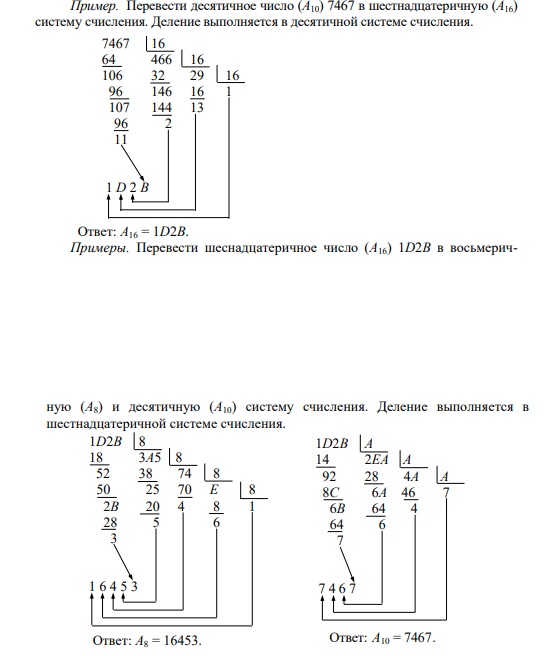
-метод подбора (обратных)степеней основания; -метод деления(умножения) на основание системы счисления

-переводы из(в) с/с с основаниями кратными 2









* **Кодирование чисел (прямой, обратный, дополнительный код).**

Кодирование чисел позволяет заменить операцию арифметического вычитания операцией алгебраического сложения с помощью двоичного сумматора. Для кодирования знака числа используется специальный двоичный разряд, называемый знаковым. При этом знак плюс кодируется двоичной цифрой 0, а минус – цифрой 1 (для системы счисления с основанием r – цифрой r − 1). Для машинного представления чисел используют три основных вида кодов: прямой, обратный и дополнительный. Общая схема кода числа: код знака . код числа.

1. **Прямой код числа**. При этом способе кодирования чисел кодируется только знак числа, а значащая часть остается без изменения
2. **Дополнительный код числа.** Число ' A называется дополнением к числу А, если выполняется соотношение: ' n A A r + = для целых чисел или ' 0 A A r + = для дробных чисел, где n – количество цифр в записи числа A.
3. **Обратный код числа**. Обратный код двоичного числа является инверсным изображением числа, в котором все разряды исходного числа принимают инверсное (обратное) значение.

***Код. Чисел.***

-прямой;

-доп. Код (Число А' называется *дополнением* к числу А, если выполняется соотношение: А + А′ = rn  для целых чисел или А + А'=r0)

-обратн. Код.

* **Переполнение разрядной сетки. Причины и признаки переполнения. Модифицированные коды**

При выполнении некоторых арифметических операций может возникнуть ситуация, когда старшие разряды результата операции не помещаются в отведенной для него области памяти. Эта ситуация называется переполнением разрядной сетки формата числа. Причиной переполнения может служить, например, суммирование двух чисел с одинаковыми знаками, которые в сумме дают величину, большую или равную единице (при сложении правильных дробей), или величине r n (при сложении целых чисел). При выполнении операций над числами с разными знаками переполнение не возникает.

Для обнаружения переполнения можно использовать следующие признаки: − знаки слагаемых не совпадают со знаком суммы; − есть перенос только в знаковый (П1) или только из знакового (П2) разряда. Если при сложении чисел с фиксированной запятой возникло переполнение, то вырабатывается сигнал переполнения разрядной сетки и вычисления прекращаются

* **Формы представления чисел в ЭВМ  (ф.з. и п.з).**

Существуют два основных способа представления данных в ЭВМ: с фиксированной и плавающей запятой (точкой)

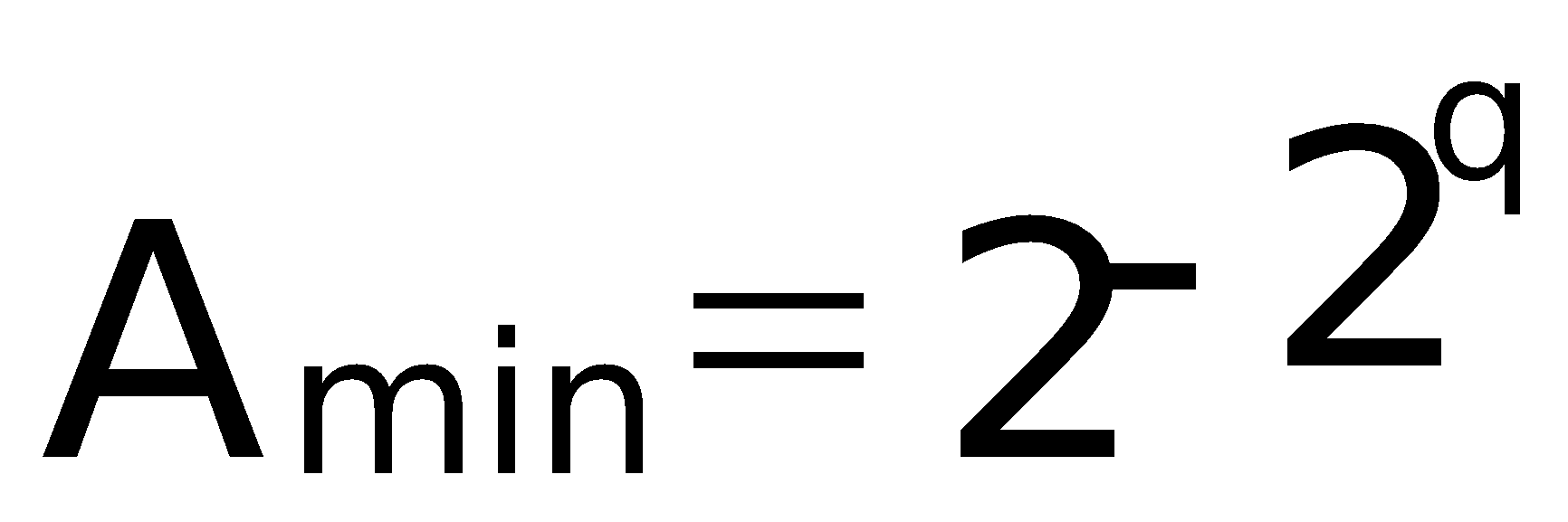
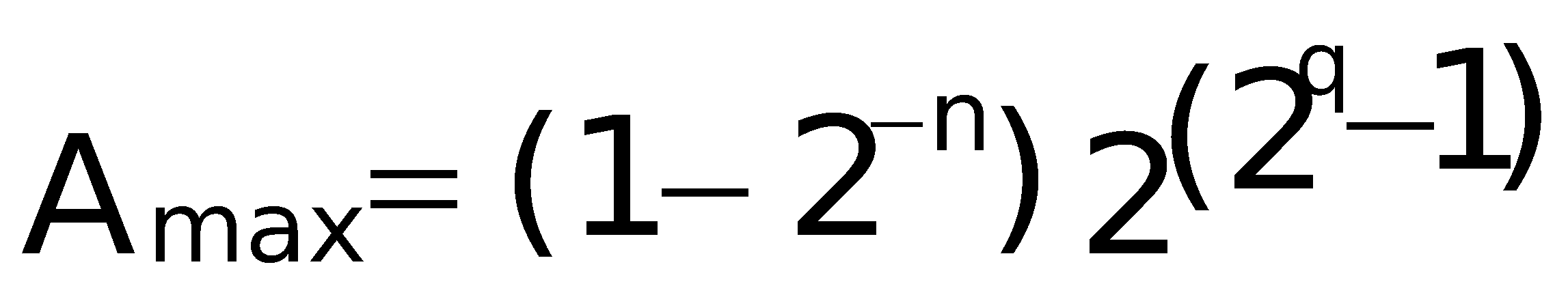
-Числа с фикс. запятой

|  |  |  |
| --- | --- | --- |
|  | целая часть | дробная часть |

Аmax=(2k-1)+(1-2-l),

-числа с плавающ. Запятой

|  |  |  |  |
| --- | --- | --- | --- |
|  | мантисса mA |  | порядок p |



* **Округление чисел.**

Рассмотрим округление только дробных чисел, целые не округляются. Так как в ЭВМ используются числа с конечным числом разрядов, а также не редко выполняются операции приведения данных одной размерности к данным другой, то операция округления выполняется часто.

Ar=[ma]rn+([A0]rk-n)-ушло за сетку.

В зависим. От учитывания A0:

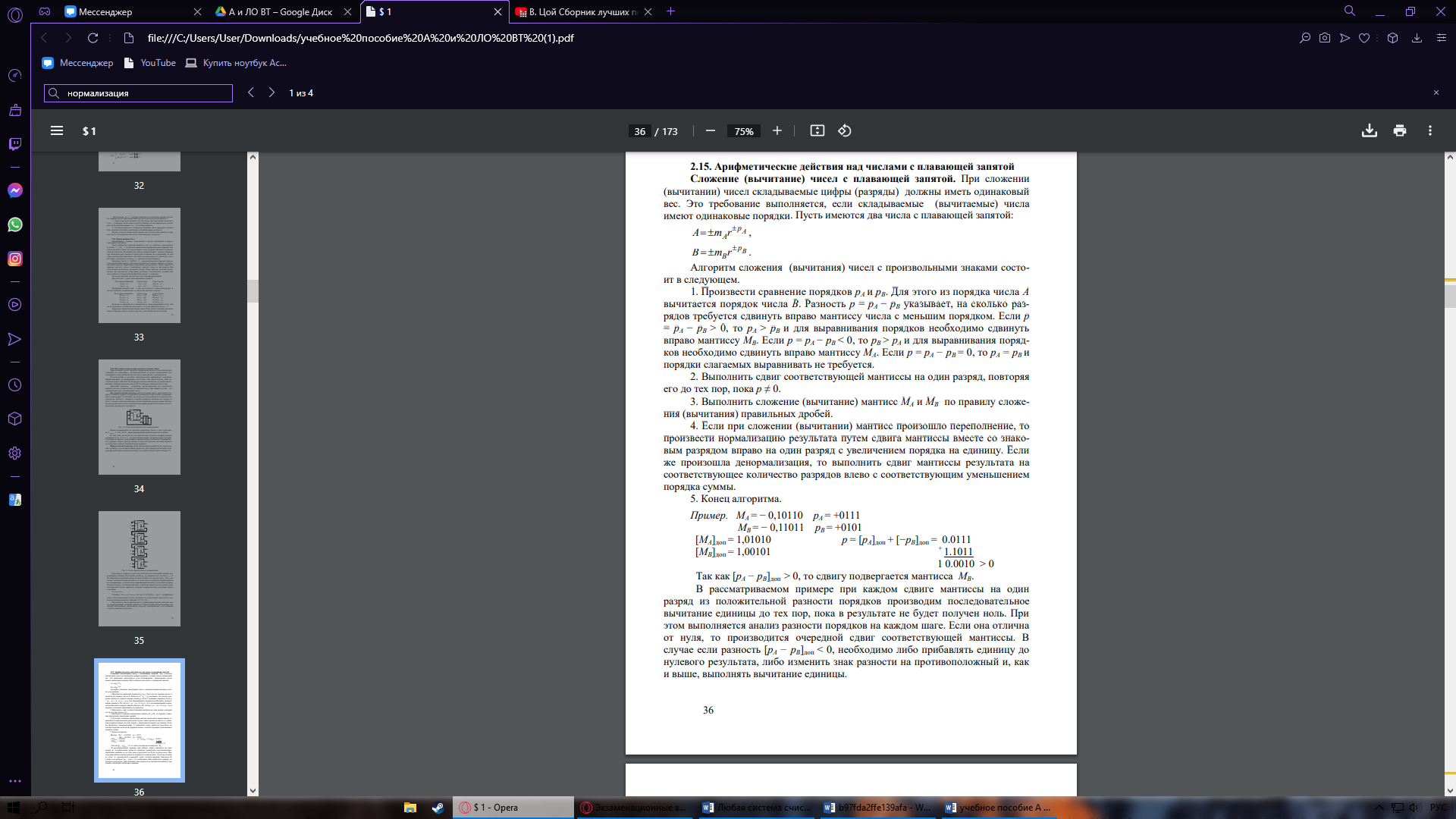
-Отбрасывание A0

-симетричное округление(При условии |А0|≥r-1 единица добавляется к младшему разряду мантиссы.Данный способ округления наиболее часто используется на практике.)

* **Сложение чисел с плавающей запятой. Нормализация чисел.**

Нормализация – процесс, относящийся к числам, записанным в форме с плавающей запятой

При сложении (вычитании) чисел складываемые цифры (разряды) должны иметь одинаковый вес. Это требование выполняется, если складываемые (вычитаемые) числа имеют одинаковые порядки



*(Целые* **1)Мн>0,Мт>0** как в прямых. **2)Мн>0,Мт<0.** [Мт]доп=2n–Мт. Так как сомножители имеют разные знаки, то произведение Мн∙Мт<0, следовательно, [Mн∙Мт]доп=22n-Mн∙Мт.Однако при умножении Мн∙[Мт]доп получается Mн∙(2n-Mт) =2n Mн - Mн∙Мт. Следовательно, погрешность в этом случае равна Δ = 22n–Mн∙Мт–2nMн+Mн∙Мт=22n–2nMн=[–Мн]доп∙22n =[[Мн]доп]доп∙22n. **3)Mн<0,Mт>0.** см. дроби. **4)Mн<0,Mт<0.** см. дроби.)

***Сложение с плав. Запятой***

1. Р=Ра-Рв. Р>0, Ра>Рв, Мв –вправо

2. Сдвиг мантиссы, пока p!=0.

3. Сложение мантисс как прав. Дроби.

4. Нормализация при переполнении.

***Умножение***

Мн∙Мт=С=А∙В=0,а1а2…аn(b12-1+b22-2+…+bn2-n)= 0+(b1∙0,а1а2…аn)2-1+…+(bn-1∙0,а1а2…аn)2-(n-1)+(bn∙0,а1а2…аn)2-n =0+b1∙A2-1+…+bn-1∙A2-(n-1)+…+bn∙A2-n=0+bn∙A2-n+bn-1∙A2-(n-1)+…+b1∙A2-1 =(…((0+bn∙A)2-1+bn-1∙A)2-1+…+b1∙A)2-1 (А)

Мн∙Мт=С=А∙В=0+bn∙A+bn-1∙A∙22+…+b1∙A∙2n-1  (Б)

Мн∙Мт=С=А∙В=(...(0+b1∙A)∙21+b2∙A)∙21+...+bn∙A) (В)

Мн∙Мт=С=А∙В=0+b1∙A∙2-1+b2∙A∙2-2+…+bn∙A∙2-n (Г)

Мл.Мт(А-Сумм(вправо);Б-произ(влево));

Ст.Мт(В-сумм(влево);Г-произ(вправо)).

***Умножение чисел с хранением переносов***

Только А. Время, затрачиваемое на сложение двоичных чисел, состоит из времени, необходимого для поразрядного сложения, и времени на формирование переноса. *Поразрядное сложение является элементарной операцией, и время на эту операцию может быть сокращено путем использования более быстродействующих элементов. В то же время если исключить необходимость выполнения межразрядных переносов при сложении, то время умножения уменьшится на tпер.*

***Время Умножение на 2 разряда множителя одновременно в прямых кодах.***

Появление любой из рассматриваемых пар множителей равновероятно. Следовательно, время умножения на два разряда множителя может быть выражено следующим соотношением: T= ( n/2 + 1) [0,75∙(tсл + tсдв) + (0,25∙tсдв], где n – количество разрядов множителя.

***Умножение в доп. Кодах.***

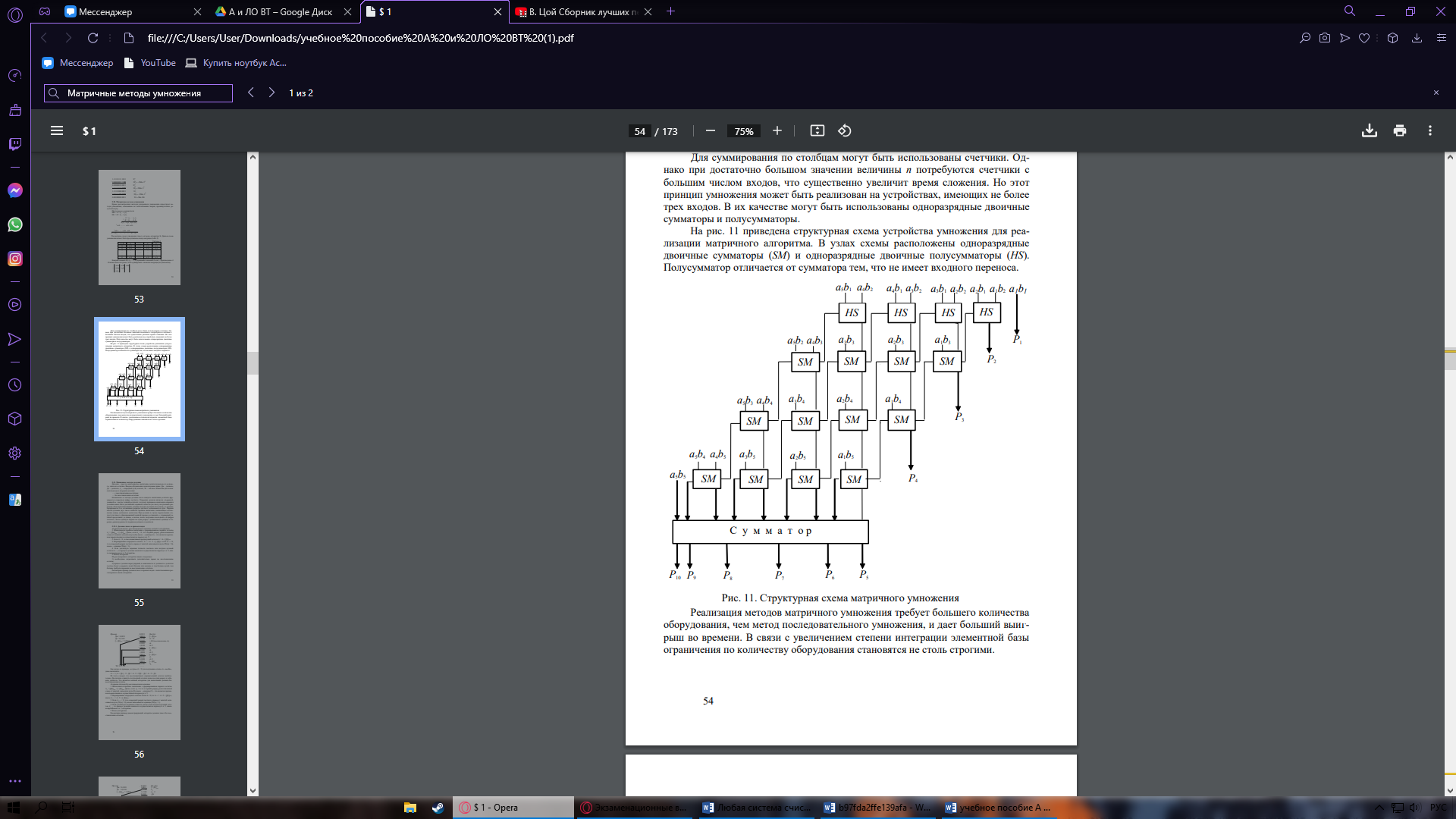
*Дробные.*  **1)Мн>0,Мт>0** Как в прямых. **2)Мн>0,Мт<0.**

[Мн∙Мт]доп=2-Мн∙Мт. Мн∙Мт′=Мн∙(1-Мт)=Мн-Мн∙Мт. Таким образом, погрешность Δ умножения равна разности [Мн∙Мт]доп. и Мт∙Мн′ Δ=2-Мн∙Мт-Мн+Мн∙Мт = 2-Мн = ([-Мн]доп)=[[Мн]доп]доп *при переполнении нужна коррекция(и знака в знач. Часть +знак меняется)* **3)Mн<0,Mт>0.** Тоже самое неудобно. ***(НА Г)*** Мн∙Мт = А ∙ В = [ A ∙ b1∙ 2-1 ]доп + [ A ∙ b2∙ 2-2 ]доп+ ... + [A∙ bn∙ 2-n ]доп На основании теоремы, доказывающей что сумма дополнительных кодов есть дополнительный код суммы, получаем [ A ∙ b1∙ 2-1 + A ∙ b2∙ 2-2 + ... + A∙ bn∙ 2-n ]доп =[Мн ∙Мт]доп. В этом случае поправка вводится автоматически на каждом этапе умножения. **4)Mн<0,Mт<0.** Mн∙Mт=2-[Mн∙Mт]доп [Mн]доп∙(1-Mт)=[Mн]доп -[Mн]доп ∙Mт=[Mн]доп - [Mн∙Mт]доп  Δ=2 - [Mн∙Mт]gдоп - [Mн]доп + [Mн∙Mт]доп = 2 - [Mн]доп = [[Mн]доп]доп

* ***Матричные методы умножения***

Кроме рассмотренных методов ускоренного умножения существуют методы умножения, основанные на использовании матриц промежуточных результатов

Строим матрицу по алгоритм. Произведение=суммирование по диогонали.



* **Машинные методы деления. Структурная схема операционного устройства выполняющего операцию деления.**

Деление – простое многократное вычитание делителя вначале из делимого, затем из остатков. Введем обозначения, используемые ниже: Дм – делимое, Дт – делитель, Аi – очередной (i-й) остаток, Чт – частное. Известны два основных подхода к операции деления: − с восстановлением остатков; − без восстановления остатков.

Конец деления после достижения точности, так же при получении нулевого остатка(тогда в оставшиеся разряды частного запис. Нули).1 является пробное вычитание, выявляющее соотношение между Дм и Дт. При делении в случае переполнения следует: числа с фиксированной запятой процесс остановить, с плавающей запятой продолжить до конца, после получения последней n-й цифры частного, число сдвигается вправо на один разряд с добавлением единицы к порядку, равному разности порядков делимого и делителя.

*Деление в прямых кодах с восстановлением остатка*

1. A1=[Дм]доп+[-Дт]доп  А<0, слева от запятой 0. иначе кон.

2. Если Аi<0, то востонавл. Пред. Остаток.

3. сдвиг влево.

4. +[-Дт]доп  знаков. цифра запис. В ответ с инверсией.

5. повторяем пока не достиг. Точность.

*прямые без восстановления*

1. A1=[Дм]доп+[-Дт]доп  А<0, в зн. Отв. 0.

2. сдивг влево

3. если А<0, то +[Дт]доп , иначе +[-Дт]доп

4. запись за А в частное с ниверсией

5. конец если конец =)

***Деление чисел в доп. кодах***

зн чт = зн Дм (mod2) зн Дт.

основа деление без вост. Остатка.

1.если знак Дм ≠ знаку Дт, то первый остаток A1=[Дм]доп+[Дт]доп, иначе A1=[Дм]доп+[-Дт]доп. Далее формируется зн. - 0, если знак А1 ≠ знаку Дт, иначе 1

2. Если знак Аi ≠ знаку Дт, то Ai+1=Ai∙2+[Дт]доп, иначе Ai+1=Ai∙2+[-Дт]доп.

3. Если знак Аi+1 ≠ знаку Дт, то в разряд частного заносится 0, иначе - 1.

4. конец если точность.

Реализация этого метода ускорения кроме устройства управления делением требует логическую схему, осуществляющую две функции:

1) сдвиг модулей делителя и делимого до тех пор, пока у модуля делителя после запятой не останется ни одного нуля;

2) выявление остатков вида 0,0…01 или 1,1…10. Степень сложности логической схемы определяется количеством разрядов, участвующих в косвенном сравнении модулей делителя и делимого (остатков).

***BCD Сум. С одинаковыми знаками.***

возможны следующие особенности:

-наличие разрешенных и запрещенных комбинаций, свидетельствующих о правильности результата или необходимости его коррекции;

-при сложении тетрад возможен потетрадный (16 единиц), а не поразрядный (10 единиц) перенос, что также требует корректировки результата.

Возможны следующие два случая сложения чисел в BCD-коде +3:

1) a + b ≤ 9 , [ ( a + 3 ) + ( b + 3 ) ] ≤ 15 и, следовательно, в тетраде суммы будут лишние 6 единиц. Чтобы тетрада суммы осталась тоже с избытком 3, нужно вычесть 3;

2) a + b ≥ 10; [ ( a + 3 ) + ( b + 3 ) ] ≥ 16. Здесь во всех случаях возникает шестнадцатеричный перенос, вместе с которым тетраду суммы покинут и шесть избыточных единиц. Чтобы тетрада суммы осталась с избытком 3, надо добавить 3.

Если складываются числа с разными знаками, то избыток в тетраде суммы будет равен нулю и суммирование, таким образом, сводится к правилам суммирования в BCD-коде.

***BCD Сум. С разными знаками.***

Представляется в обратных кодах.

1)(a-b)>=0, При образовании инверсии отрицательной тетрады в нее добавляются 15 единиц. Эти 15 единиц находятся и в сумме. Но благодаря шестнадцатеричному переносу из тетрады уходит 16 единиц (15+1 − эта единица восстанавливается добавлением по цепи циклического переноса ).

2)(a-b)<0, Здесь, как и в предыдущем примере, в тетраде суммы пятнадцать лишних единиц. Но при переходе от инверсной формы к прямой лишние единицы уничтожаются сами собой. Это то же самое, что от значащей части суммы вычесть пятнадцать: 1011 - 1111 = 0100.

Если число + и тетрада 1111, то добавим 1010, если число – и тетрада 0000, добавим 0110.

***BCD c избытком 3.***

Самодополняющийся, но не взвешенный.

1)(a+b)<=9 [ (a+3) + (b+3) ] <=15. Лишние 6 еденицы, что бы тетрада суммы была с избытком 3, вычтем 3(+1100)

2)(a+b)>=9 [ (a+3) + (b+3) ] >=16. Возникает 16-ричный перенос, 6 изб. Едениц тоже покинут тераду. Надо добваить 3(+0011)

Если складываются числа с разными знаками, то избыток в тетраде суммы будет равен нулю и суммирование, таким образом, сводится к правилам суммирования в BCD-коде.

*Правило*. Если из тетрады был перенос, надо добавить +0011, если переноса не было, – 0011 (добавить 1100), независимо от знака слагаемых и знака суммы.

***Основные законы АЛ***

Основные законы АЛ позволяют проводить эквивалентные преобразования логических функций, записанных с помощью операций И, ИЛИ, НЕ, упрощать запись.

x + 1=1; x + x=1; x ∙ 0=0; x ∙ x=0; x + x + … + x=x;

x∙x∙…∙x=x. Так же работают переместительный, сочетательный и распределительный(диз. Кон.) законы для диз. Кон и мод2. Правило де Моргана x1+x2=x2 ∙ x1; x1∙x2=x2+x1

**Основные понятия алгебры логики.**

Это двоичные переменный и переключательные (булевы) функции. Принимают 1(0). Двоичные переменные являются аргументами булевых функций. *Булева функция – это функция, и аргументы и значение которой принадлежат множеству{0,1}.* Алгебра <B;Ω>, где Ω – множество всевозможных булевых функций, называется алгеброй логики (булевой алгеброй).

Переменная *xi* называется *фиктивной* (несущественной) переменной функции *f*(*x*1*,···,xn*), если *f*(*x*1*,···,xi-*1,0*,xi+*1*,···,xn*) *= f*(*x*1*,···,xi-*1,1*,xi+*1*,···,xn*) для любых значений *x*1*,···,xi-*1*,xi+*1*,···,xn*.Иначе переменная *xi* называется *существенной*.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| х1х2 | f0 | f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | F10 | f11 | f12 | f13 | f14 | f15 |
| 00 01 10 11 | 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 |

фf0 (x1, x2) = 0 - тождественный ноль (константа 0);

f1 (x1, x2)=x1∙x2 – конъюнкция (логическое произведение, И). Иногда употребляется знак & или /\;

f3 (x1, х2) = x1 − повторениеx1;

f5 (x1, x2) = x2 − повторение x2;

f6 (x1, x2) = x1 ⊕x2 − сложение по модулю 2 или mod 2;

f7 (х1, х2) = x1 + x2 − дизъюнкция (логическое сложение, ИЛИ или V);

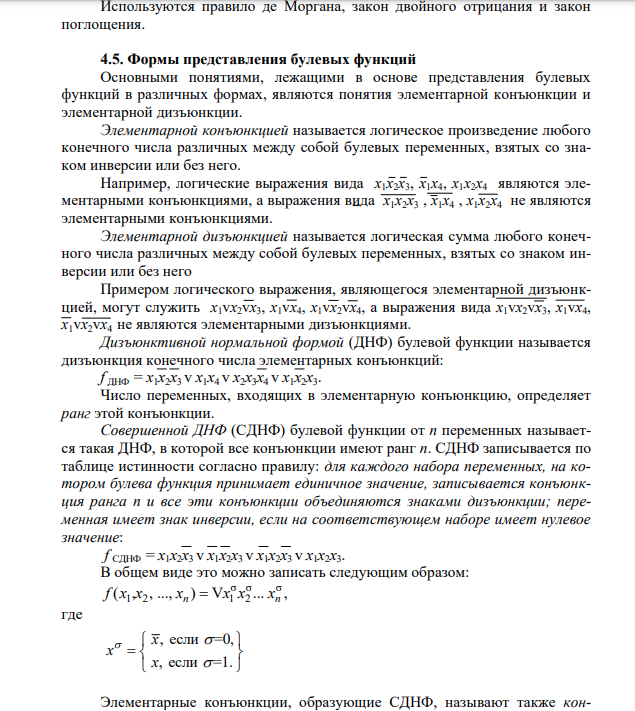
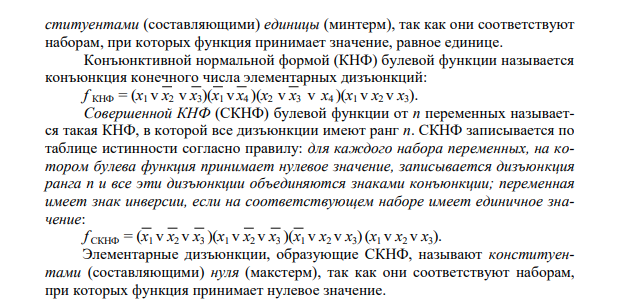
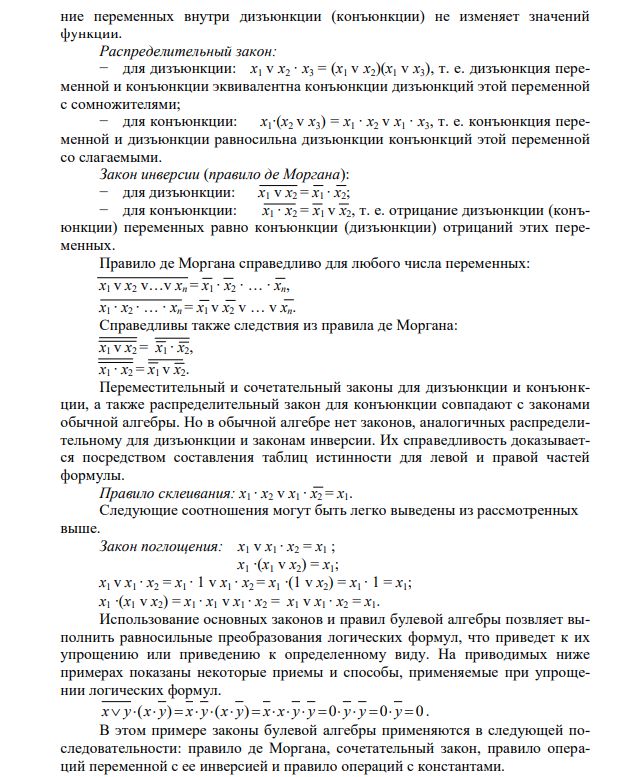
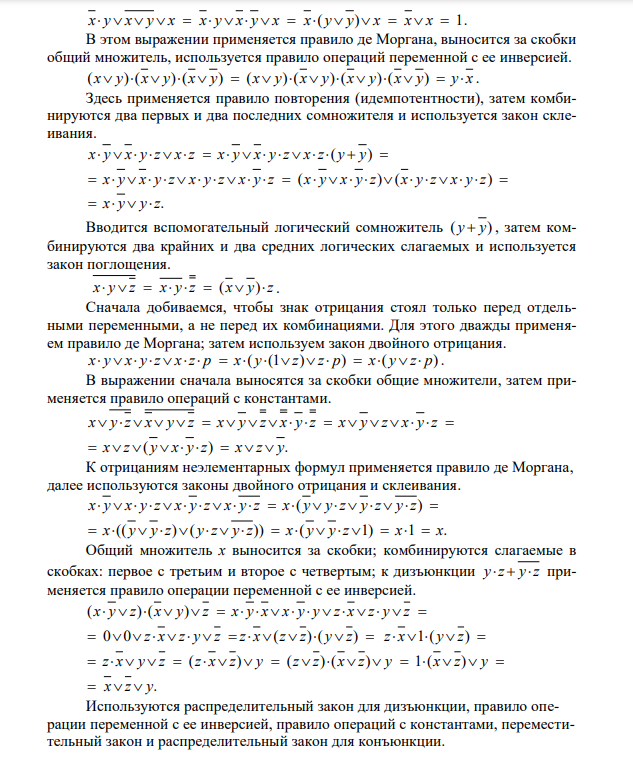
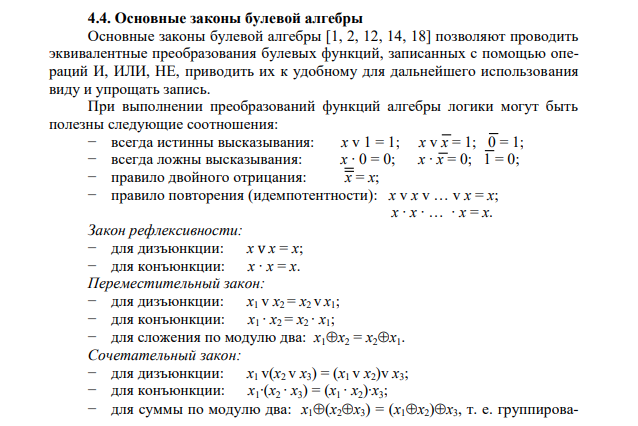
f8(x1,x2)=x1*↓*x2−функция Вебба(стрелка Пирса, ИЛИ-НЕ);

f9 (х1, х2) = x1 ~ x2 − эквивалентность;

f13(x1, x2) = x1 → x2 − импликация;

f14(x1, x2) = x1 \ x2 − штрих Шеффера (И-НЕ);

*Импликация*. Это высказывание, принимающее ложное значение только в случае, если x1 истинно, а x2 ложно



***Формы представления ФАЛ.кратко***

*Элементарной конъюнкцией* называется логическое произведение любого конечного числа различных между собой булевых переменных, взятых со знаком инверсии или без него. *Элементарн. Дизъюнкция – тоже самое.*

*Дизъюнктивной нормальной формой* (ДНФ) булевой функции называется дизъюнкция конечного числа элементарных конъюнкций.

*Совершенной ДНФ* (СДНФ) логической функции от n аргументов называется такая ДНФ, в которой все конъюнкции имеют ранг n. f СДНФ = x1x2x3 + x1x2x3 + x1x2x3 + x1x2x3. Элементарные конъюнкции в СДНФ – конституэнты еденицы(минетрмы)

Конъюнктивной нормальной формой (КНФ) булевой функции называется конъюнкция конечного числа элементарных дизъюнкций. fКНФ = (x1 + x2 + x3)(x1 + x4 )(x2 + x3  + x4 )(x1 + x2 + x3).

*Совершенной КНФ* (СКНФ) логической функции от n аргументов называется такая КНФ, в которой все дизъюнкции имеют ранг n. Элементарные дизъюнкции, образующие СКНФ, называют *конституентами* (составляющими) *нуля* (макстерм).

Чтобы получить совершенную дизъюнктивную нормальную форму, надо взять все наборы, на которых значение функции равно 1, и записать для каждого из них конъюнкцию переменных и их отрицаний. Если в наборе значение переменной равно 0, то переменную надо взять с отрицанием, если 1 – без отрицания. Из получившихся конъюнкций надо построить дизъюнкцию.

Чтобы получить совершенную конъюнктивную нормальную форму, надо взять все наборы, на которых значение функции равно 0, и записать для каждого из них дизъюнкцию переменных и их отрицаний. Если в наборе значение переменной равно 0, то переменную надо взять без отрицания, если 1 – с отрицанием. Из получившихся дизъюнкций надо построить конъюнкцию.

***Классы ФАЛ. Функционально полные наборы.***

*Функционально полной системой* *булевых функций* (ФПСБФ) называется совокупность таких булевых функций (f1, f2, ... fk), посредством которых можно записать произвольную булеву функцию f.

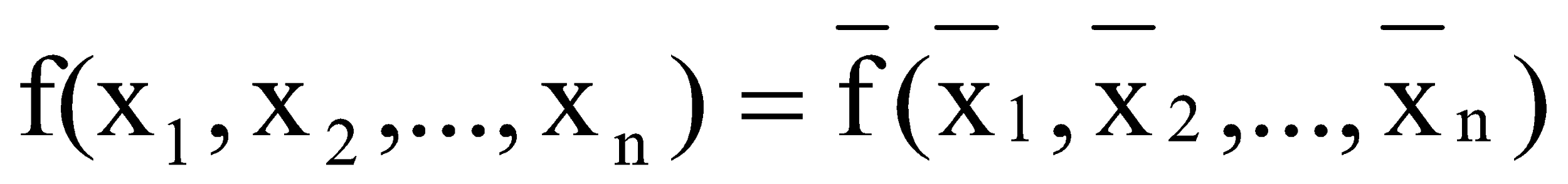
*Предполные классы булевых функций.*

*Класс линейных функций*. Функция алгебры логики называется линейной, если ее можно представить полиномом первой степени:

*Класс функций, сохраняющих ноль*. Если у них 0 на 0..0

*Класс функций, сохраняющих единицу.* Если у них f(1,...,1)=1.

*Класс монотонных функций*. Функция алгебры логики называется монотонной, если при любом возрастании набора аргументов значения этой функции не убывают.Если у двух наборов есть и большие и меньшие аргументы f(0,1) и f(1,0), то наборы считаются несравнимыми. Таким образом, если f(0,0)≥f(0,1)≥f(1,1) или f(0,0)≥f(1,0)≥f(1,1),то функция f является монотонной.

*Класс самодвойственных функций*. 

*Базисом* называется полная система ФАЛ, с помощью которой любая ФАЛ может быть представлена суперпозиций исходных функций.

*Минимальной дизъюнктивной нормальной* формой (МДНФ) булевой функции называется ДНФ, содержащая наименьшее число букв (по отношению ко всем другим ДНФ, представляющим заданную булеву функцию).

Булева функция g(x1,...,xn) называется *импликантой* булевой функции f(x1,...,xn), если для любого набора переменных, на котором g=1, справедливо f=1.

Импликанта g булевой функции f, являющаяся элементарной конъюнкцией, называется *простой*, если никакая часть импликанты g не является импликантой функции f.

* **Метод Квайна − Мак-Класки**

Этот метод ориентирован на числовое задание булевой функции в виде кубического комплекса, состоящего из 0-кубов. Метод представляет собой формализованный на этапе нахождения простых импликант метод Квайна. Основной недостаток метода Квайна – необходимость выполнения полного сравнения (склеивания) всех минтермов на этапе нахождения простых импликант. В случае большого их количества сложность этого сравнения существенно возрастает.

Для получения минимальной формы логической функции необходимо в СДНФ произвести все возможные склеивания и поглощения, и получить сокращенную ДНФ. Сокращенная ДНФ в общем случае может содержать лишние простые импликанты, которые необходимо выявить на втором этапе минимизации.

*Метод применим только к совершенной ДНФ*

***Метод квайна – Мак-класки.***

Кубы задаются числовым способом и состоят из 0-кубов. Метод представляет собой формализованный на этапе нахождения простых импликант метод Квайна. Первым шагом все исходные n-кубы разбиваются на непересекающиеся подгруппы по количеству единиц в кубе. Попарное сравнение проводим между соседними группами кубов, так как кубы, порождающие новые кубы, должны иметь кодовое расстояние 1. Полученные 1-кубы разобьем на n групп кубов в зависимости от местоположения свободной координаты в кубе. И.т.д далее Квайн.

|  |  |  |  |
| --- | --- | --- | --- |
| ∩ | 0 | 1 | х |
| 0 | 0 | ∅ | 0 |
| 1 | ∅ | 1 | 1 |
| x | 0 | 1 | х |

***Спец. Операции Рота.***

|  |  |  |  |
| --- | --- | --- | --- |
| \* | 0 | 1 | х |
| 0 | 0 | Y | 0 |
| 1 | у | 1 | 1 |
| x | 0 | 1 | х |

(\*) образуется новый r-куб, если кодовое расстояние двух исходных кубов равно 1.

(∩) выделения куба с, являющегося общей частью кубов а и b

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | x |
| 0 | z | y | z |
| 1 | y | z | z |
| x | 1 | 0 | z |

(#) служит для удаления из куба а общей части кубов а и b

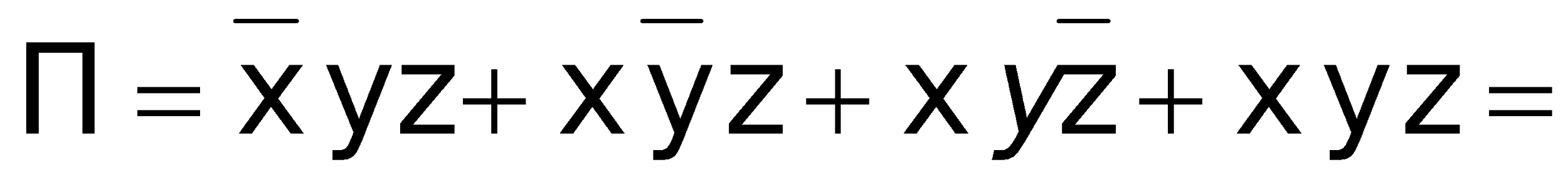
***Непосредственно Рота.***

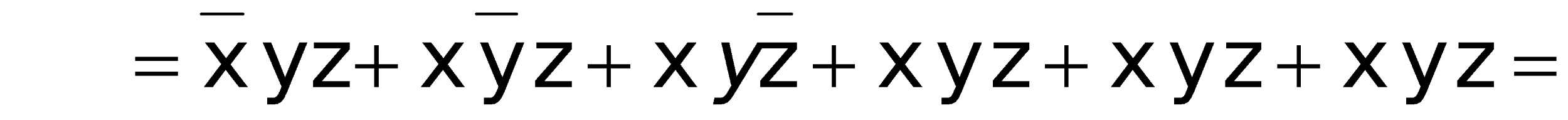
Общий алгоритм: нахождении множества Z простых импликант комплекса К; выделении L-экстремалей на множестве Z; применении алгоритма ветвления при отсутствии L-экстремалей; нахождении абсолютно минимального покрытия из некоторого множества избыточных покрытий.

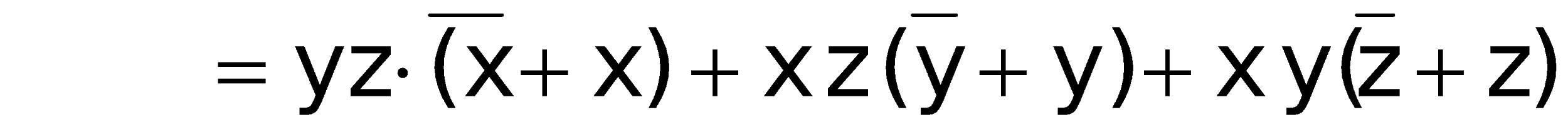
**2. определении L-экстремалей**

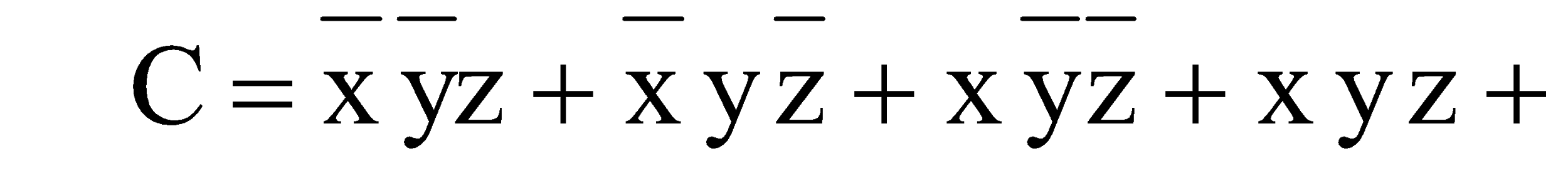
оперция # с множеством Z; операция Z ∩ L; поиск непокрытых вершин L1=L#E; Z-E ∩ L1

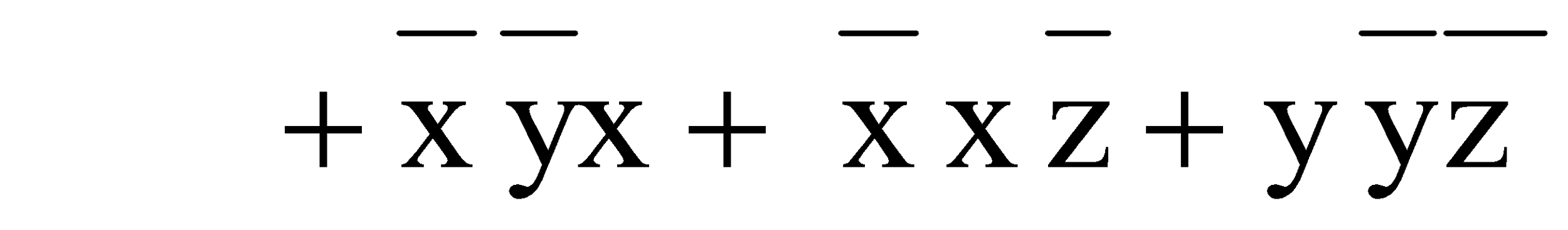
***Одноразрядного полного комбинационного сумматора***

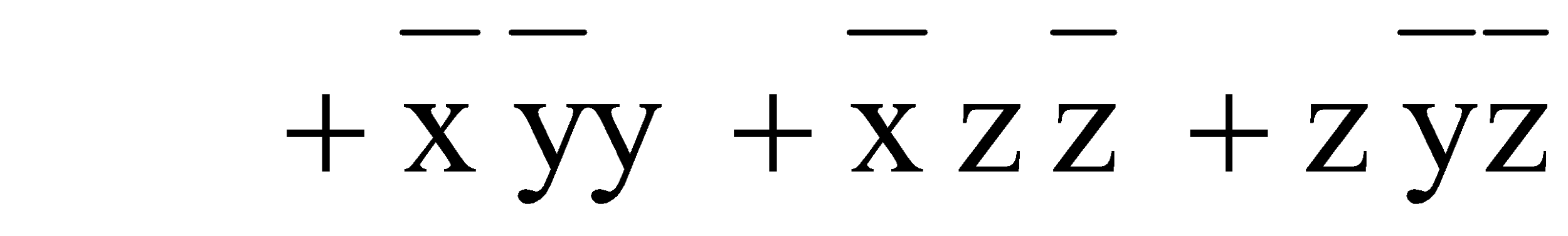


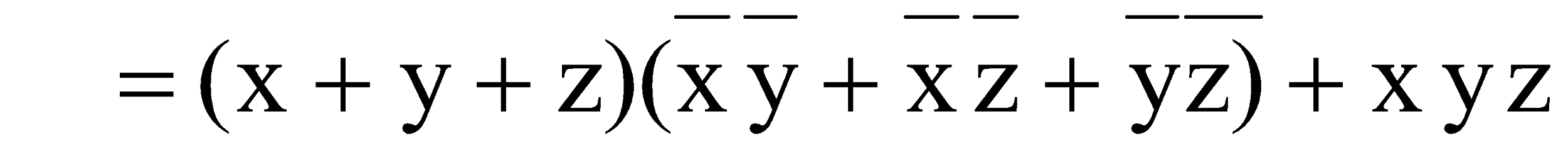


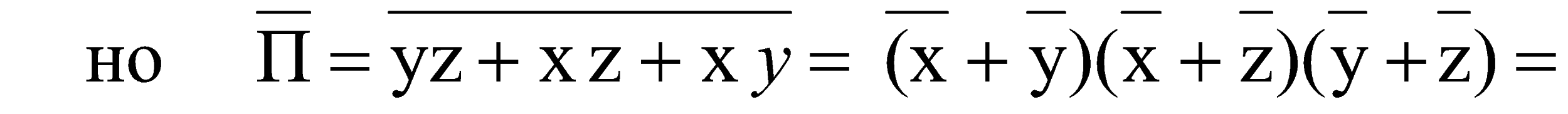


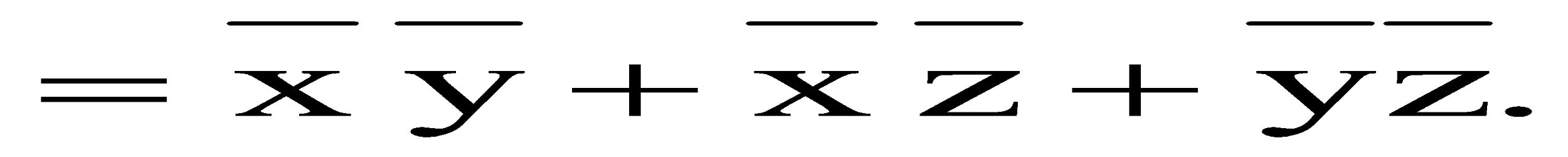




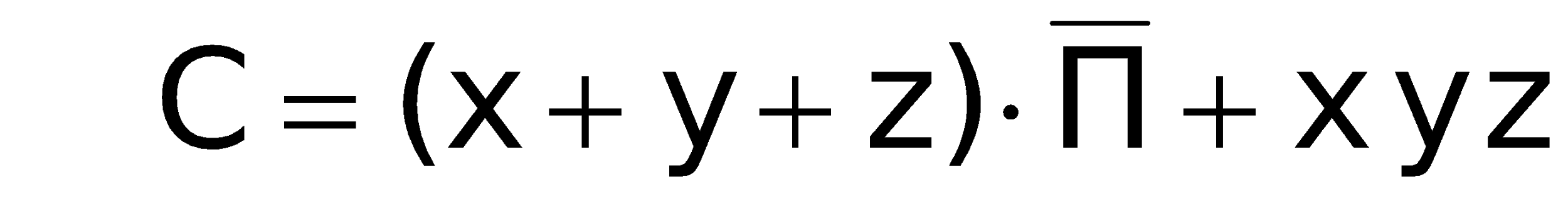




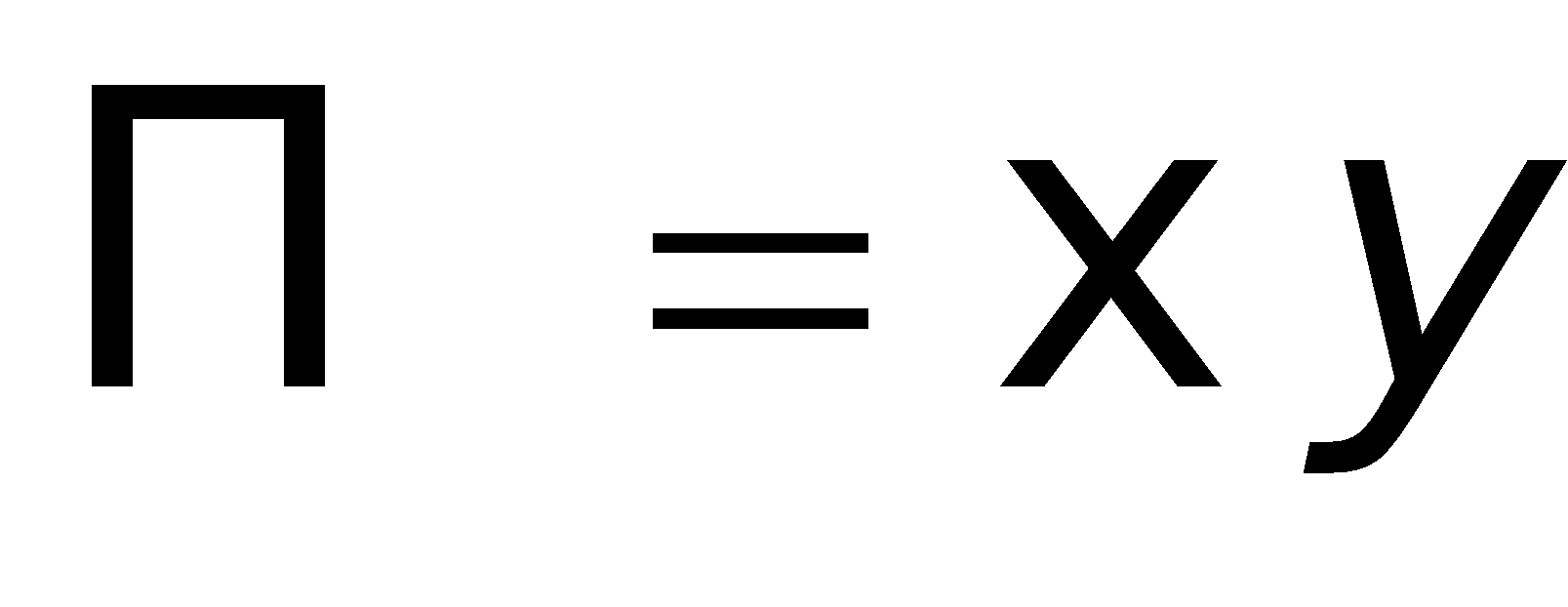
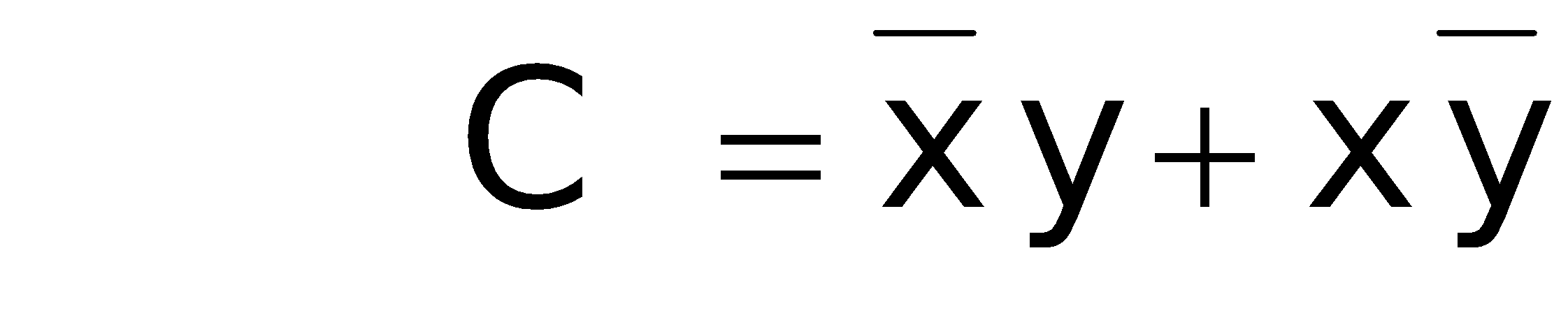


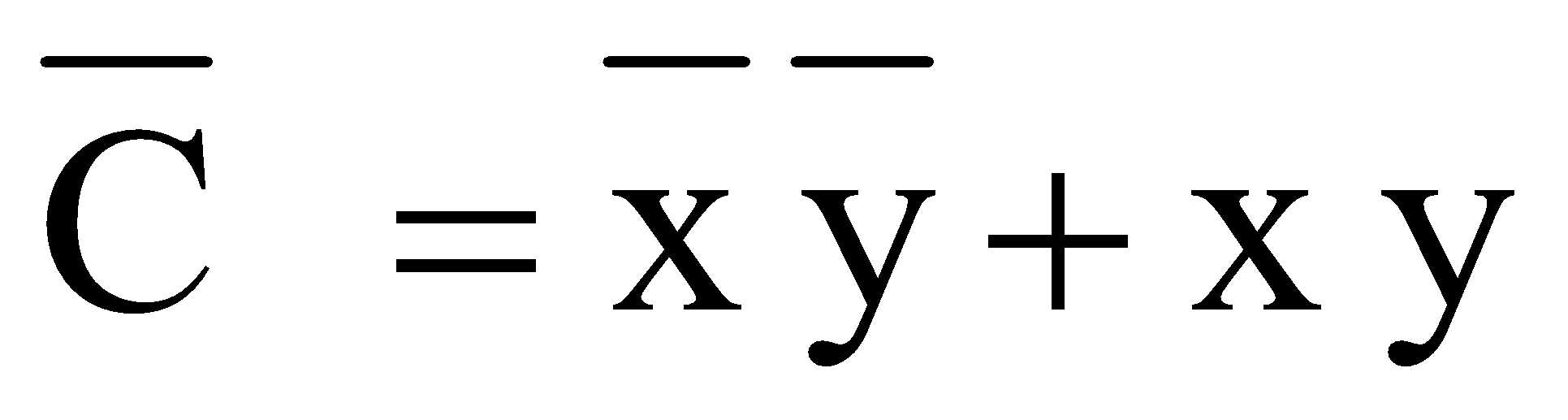
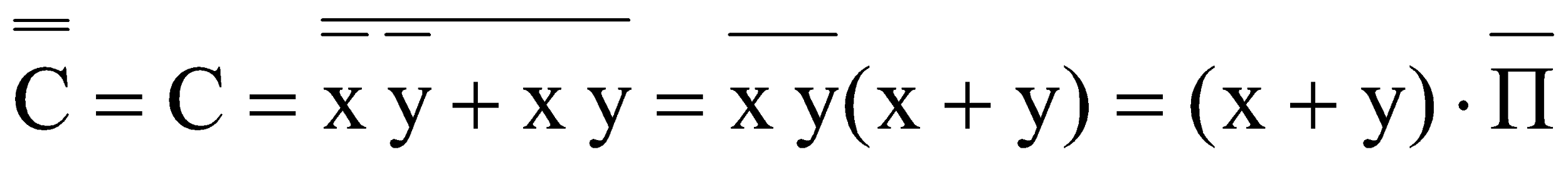


С учетом совместной минимизации функция **C** =

.

***одноразрядного комбинационного полусумматора***

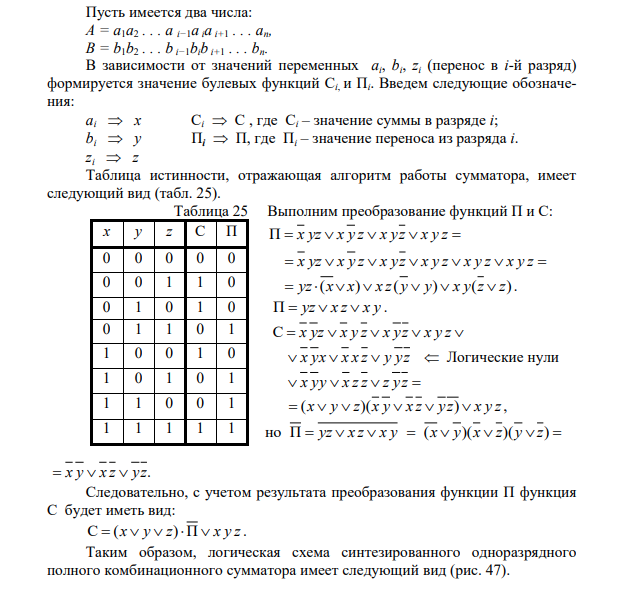
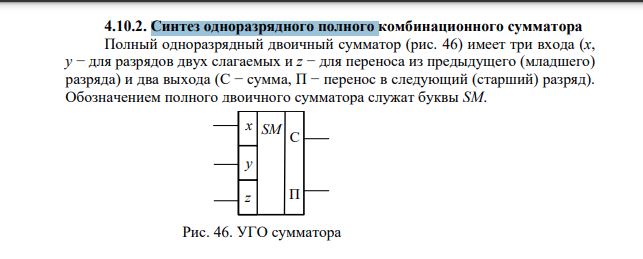
  Данная схема может быть упрощена, если функция C будет записана на нулевых наборах и использована совместная минимизация.

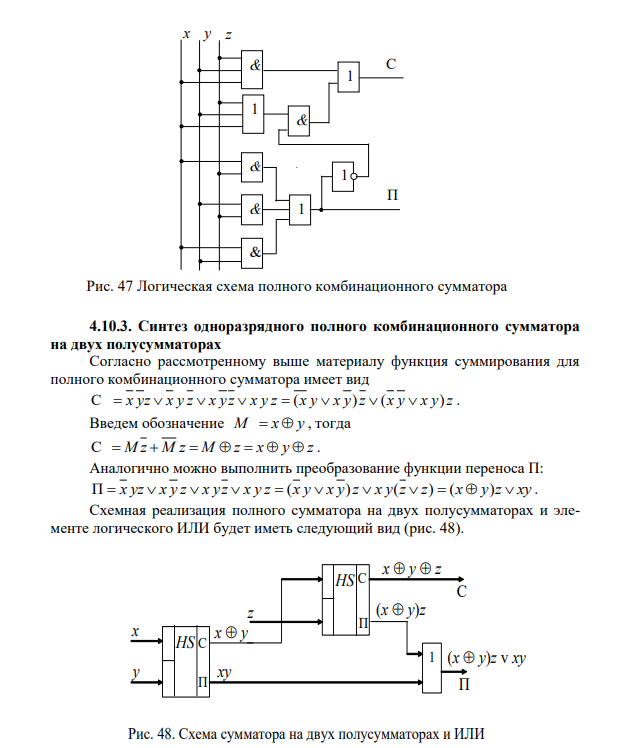
-Округление по дополнению.(что в n+1 разряде, то к n добовляем).

-Случайное округление.(ген. Случ. Чисел. 0-1)

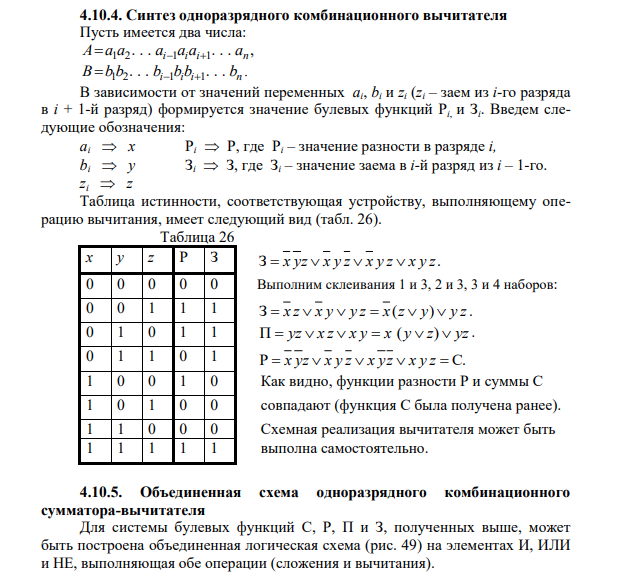
* **Синтез одноразрядного полного комбинационного сумматора**



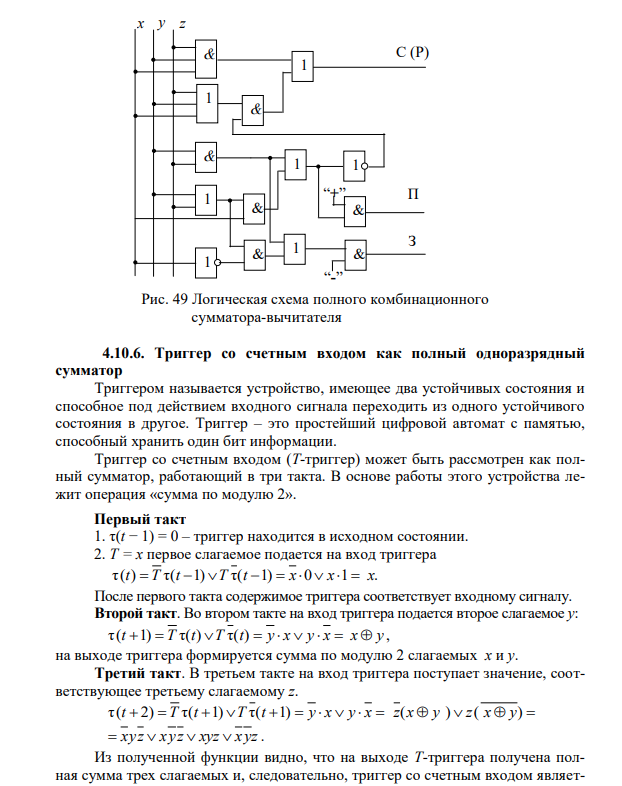
* **Синтез одноразрядного полного комбинационного сумматора на двух полусумматорах**



* **Синтез одноразрядного комбинационного вычитателя**



* **Триггер со счетным входом как полный одноразрядный сумматор**



**Гонки.** Если в течение первого полупериода (Ти) между триггерами первой ступени и возникают ”состязания”, то они все равно не изменяют состояния триггеров второй ступени, поскольку отсутствует синхросигнал Ти. Затем, с приходом синхроимпульса Ти, изменяют свое состояние триггеры второй ступени. Промежуточные коды, формируемые на их выходах, приводят к изменению (и, возможно, искажению) 1,…,r, а следовательно, и D1,…,Dr. Однако триггеры первой ступени не изменят своего состояния, поскольку отсутствует сигнал Ти. Таким образом, в итоге верный код состояния as с выходов памяти первой ступени переписывается в триггеры второго уровня, что соответствует переходу автомата в состояние as.

* **Принцип микропрограммного управления**

Принцип микропрограммного управления Устройства вычислительной техники перерабатывают информацию, выполняя над ней какие-либо операции. Функцией ОА является выполнение заданного множества операций над входными наборами c целью формирования выходных наборов, которые представляют собой результаты операций. Функциональная и структурная организация ОА базируется на принципе микропрограммного управления [17, 19], который состоит в следующем:

1. Любая операция, реализуемая устройством, рассматривается как сложное действие, которое разделяется на последовательность элементарных действий, выполняемых за один такт его работы. Эти элементарные действия называются микрооперациями.

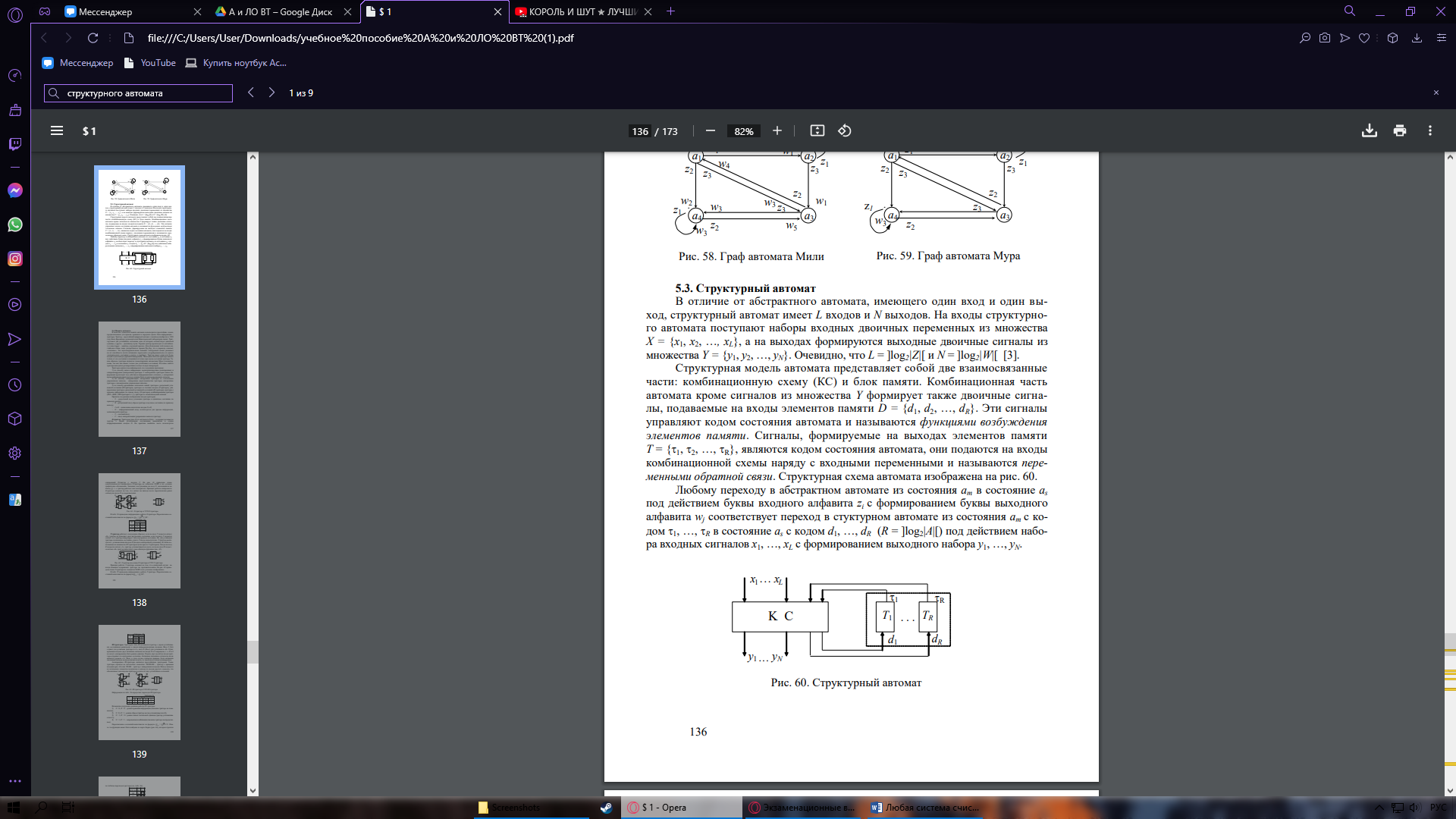
2. Для управления порядком следования микроопераций используются логические условия. Проверка значений логических условий в каждом такте работы автомата позволяет определить группу выполняемых микроопераций.

3. Совокупность операций, выполняемых за один такт работы автомата, называется микрокомандой.

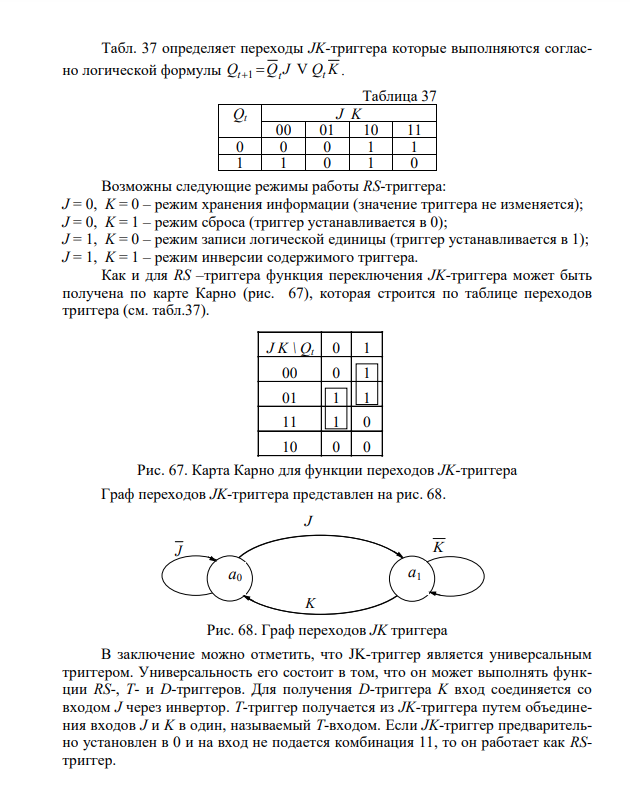
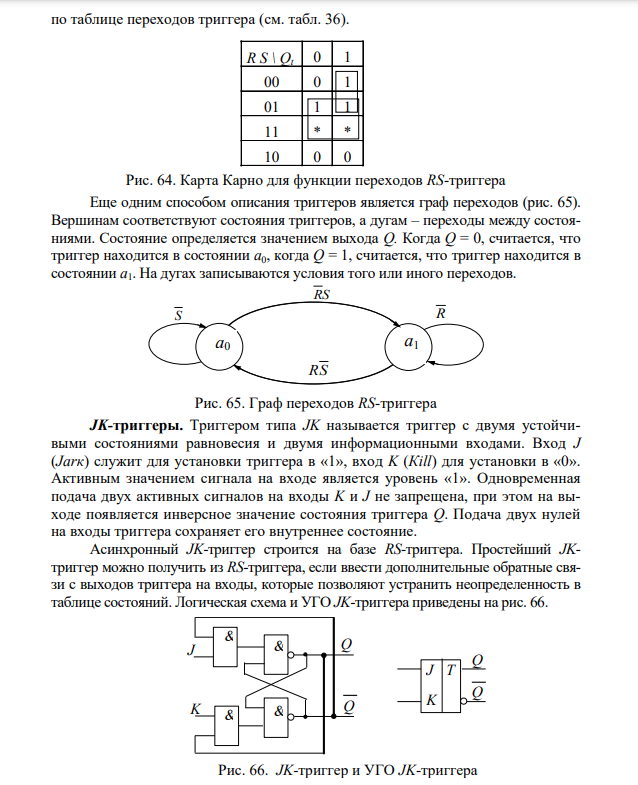
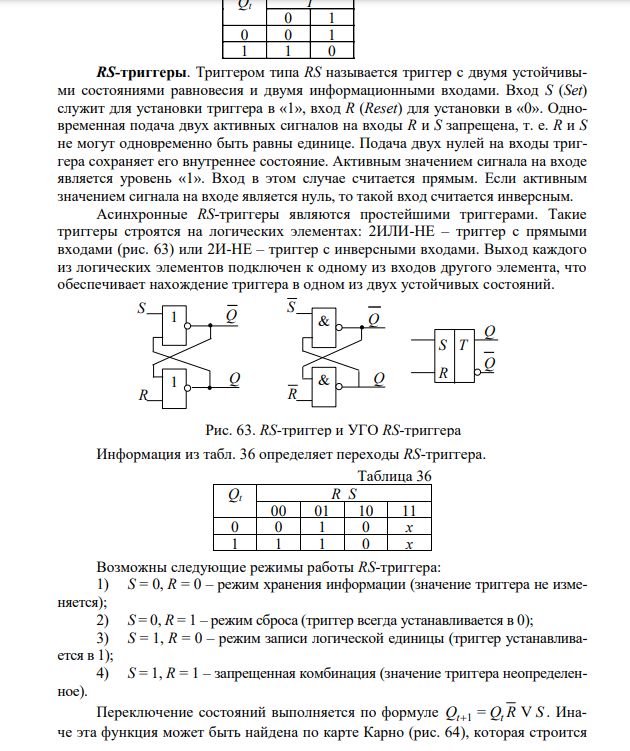
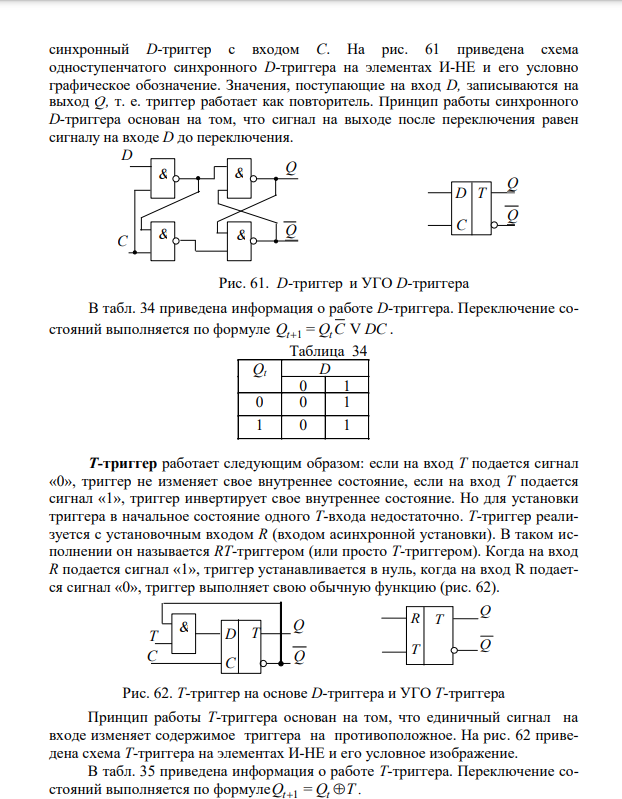
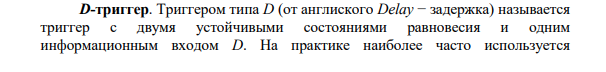
4. Процесс выполнения микрокоманд в устройстве описывается в форме алгоритма, который представляется в терминах микроопераций (микрокоманд) и логических условий и называется микропрограммой. Микропрограмма определяет порядок проверки логических условий и выполняемых микроопераций (микрокоманд) для получения требуемых результатов.

5. Принцип, согласно которому алгоритм работы некоторого устройства описывается в перечисленных выше терминах, называется принципом микропрограммного управления. Таким образом, из принципа микропрограммного управления следует, что структура и порядок функционирования операционных устройств определяется алгоритмом выполнения операции. К элементарным действиям над словами информации (микрооперациям) относятся: передача информации из одного регистра в другой, взятие обратного кода, сдвиг и т. д. Конечный автомат, алгоритм работы которого может быть описан на основе принципа микропрограммного управления, называется микропрограммным автоматом (МПА).

* **Структурный автомат**



* **Память автомата D-,.T-, RS-, JK-триггеры..**



* **Граф-схема алгоритма**

