МІНІСТЕРСТВО ОСВІТИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

«Комп'ютерні системи»

КИЇВ НТУУ «КПІ» 2012

«Основні підходи до підвищення продуктивності обчислювальних систем»

Будемо розрізняти 2 підходи:

1) Підвищення продуктивності за рахунок збільшення частотних можливостей елементної бази.

2) Підвищення продуктивності за рахунок збільшення кількості елементів.

1) $P = 10^{12}\ flops$ (продуктивність).

$\nu = 10^{14}$ 1/с (тактова частота)

$T = 10^{-14}$ с (час переключення)

$V_{\text{пр}} = \frac{c}{3} = \frac{3*10^8}{3} = 10^8\ \frac{\text{м}}{\text{с}}$ (швидкість розповсюдження сигналу у дроті)

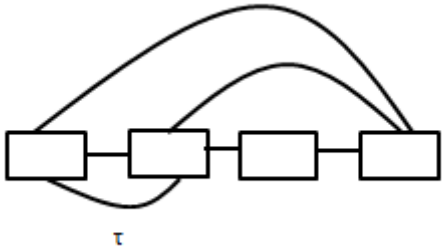


Рис.1

$$L_{\text{пред}} = V * t * \frac{1}{\gamma} = \frac{1}{\gamma} * \frac{C}{3} * \frac{1}{\nu} = \frac{1}{10^2} * \frac{3 * 10^8\ \frac{\text{м}}{\text{с}}}{3} * \frac{1}{10^{14}\ \frac{1}{\text{с}}} = 10^{-8}\ \text{м} = = 10^{-2}\ \text{мкм},$$

$$\gamma = 100$$

(максимальна довжина дроту, яка не буде порушувати частотний баланс системи)

Теоретична межа тактової частоти дорівнює 10^{18}

Можливості першого підходу виключати не можна, але вони практично вичерпані.

2) $P = K * \nu * S$, ν – частота, S – кількість елементів.

$$S = p * V_{\text{пр}} = p * \left(\frac{1}{\gamma} * \frac{c}{3} * \frac{1}{\nu}\right) = a * \frac{1}{\nu^3},$$ $p - const$ для данної элементної бази

Якщо зменшити тактову частоту на порядок, створюються передумови для підвищення кількості елементів на 3 порядки.

$$P = K * \nu * S = K * a * \frac{1}{\nu^3} * \nu = A * \frac{1}{\nu^2}$$

При зменшенні частоти створюються передумови для збільшення продуктивності на 2 порядки.

Гіпотеза Мінського.

М. Мінський сформулював гіпотезу: продуктивність паралельної системи зростає (приблизно) пропорційно логарифму числа процесорів - це набагато повільніше, ніж лінійна функція.

$$P = log_2 N - \text{частота Мінського}$$

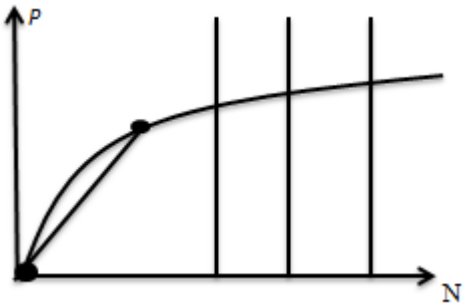


Рис.2 Нелінійний характер збільшення Р.

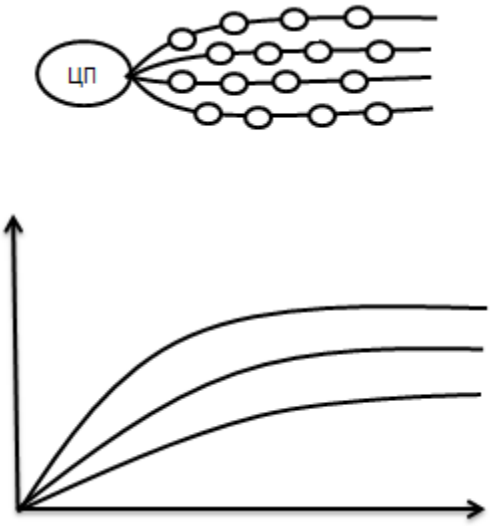


Рис. 3 Нелінійний характер збільшення

Системи можуть бути ізоефективними.

Всі регістри загального призначення можна розглядати як деяку надоперативну пам'ять, тим самим створюються передумови для мінімізації операція взаємодії з ОП та виконання кількох команд на основі операцій типу «регістр-регістр».

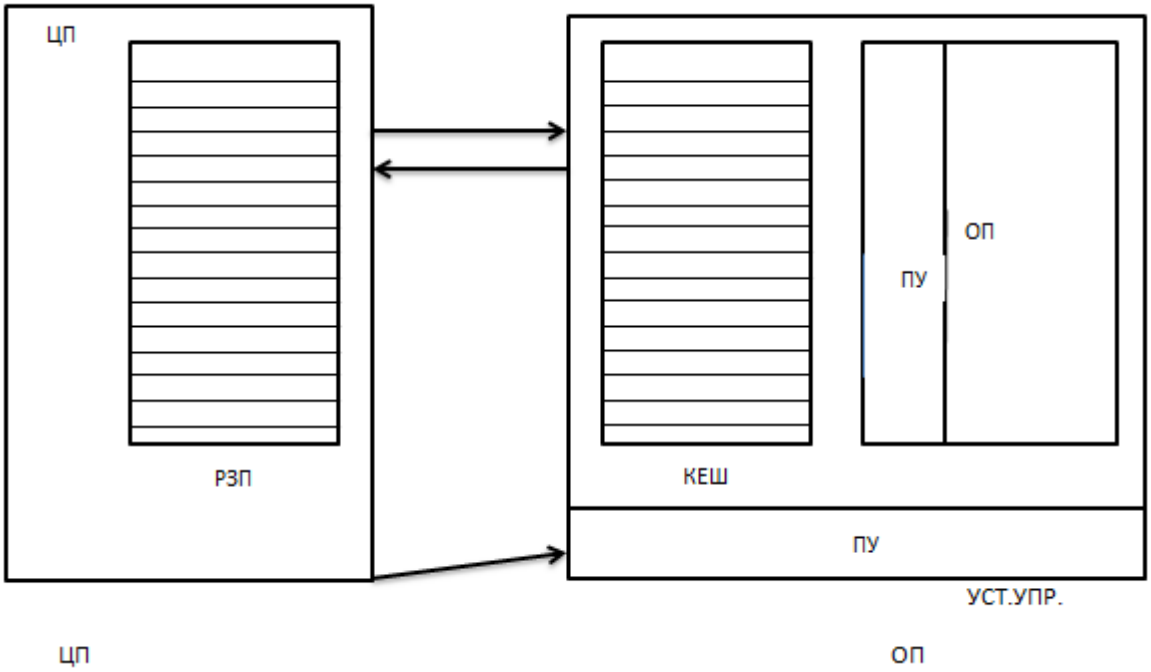


Рис. 4

З РЗП інформацію може вибирати тільки ЦП. Буферна пам'ять належить множині пристроїв управління.

У даній машині можна широко використовувати операцію групового пересилання (якщо немає кеш) з ОП в РЗП.

Завдяки РЗП з'являється можливість використовувати безліч форматів команд, оскільки обсяг пам'яті РЗП невеликий і формат команди розростається в розумних межах.

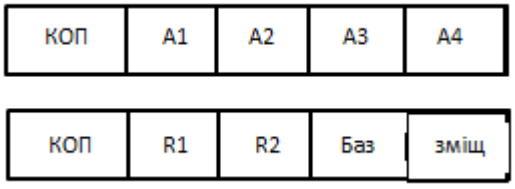


Рис. 5

Загальна структурна організація ЕОМ різних структурних поколінь

ОС розвивалися за 2 напрямками:

1) Ускладнення алгоритмів функціонування системи і рішення задач мультипрограмування.

2) Збільшення безлічі елементів системи і рішення задач мультиобробки.

1) Структурне покоління

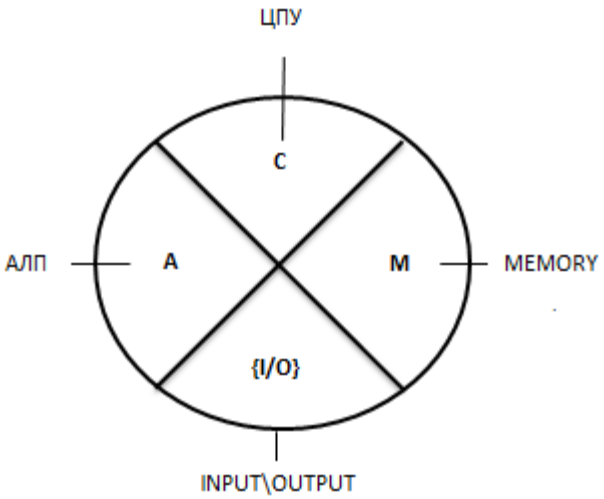


Рис. 6 Структурна організація ЕОМ 1-ого покоління

Єдине центральний пристрій управління, управляє усіма структурними елементами машини.

Усі структурні елементи машини не автономними. Єдиний пристрій не підлягає ніяким змінам.

2) Структурне покоління

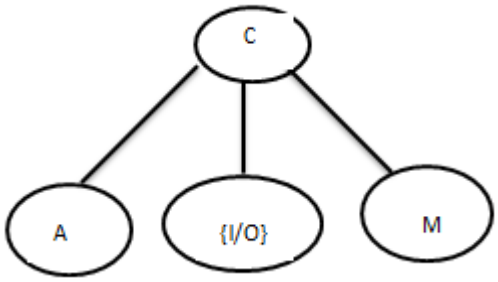


Рис. 7 Структурна організація ЕОМ 2-ого покоління

Всі елементи автономні. Необхідні умови для спільної роботи різних пристроїв в часі вже мали місце, але не було достатніх умов.

Основна проблема машин 2 структурного покоління полягало в тому, що б якимось чином визначити моменти завершення різних структурних елементів, з метою їх подальшого завантаження. При відсутності системи переривання її функції можна було б замінити.

1) Регулярним опитуванням стану цих елементів за допомогою центр. У, але в цьому випадку велику частину часу ПУ буде зайнято саме цими допоміжними функціями, що істотно сповільнить роботу даної машини.

2) В якості деякої системи переривання може використовуватися сам програміст. Це означає, що програміст відмовляється від стандартних програм введення-виведення і вирішує питання програмування введення-виведення безпосередньо в рамках своєї прикладної програми.

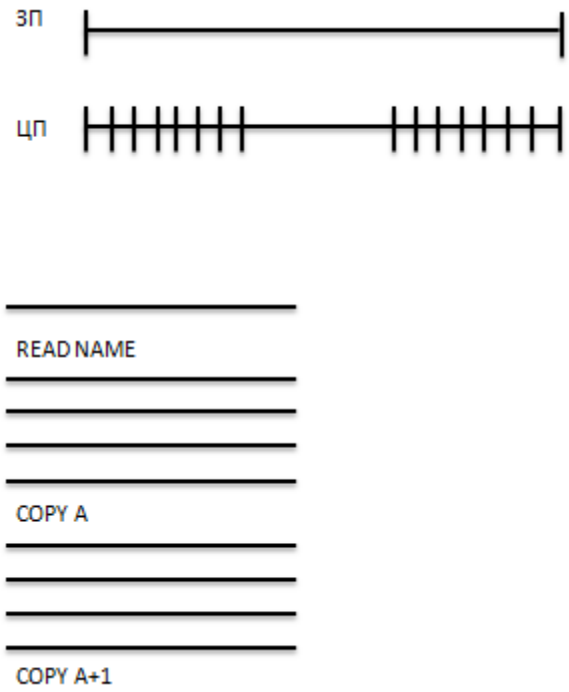


Рис. 8

Мультипрограмування - це одночасне виконання прикладної програми і безлічі програм вводу / виводу.

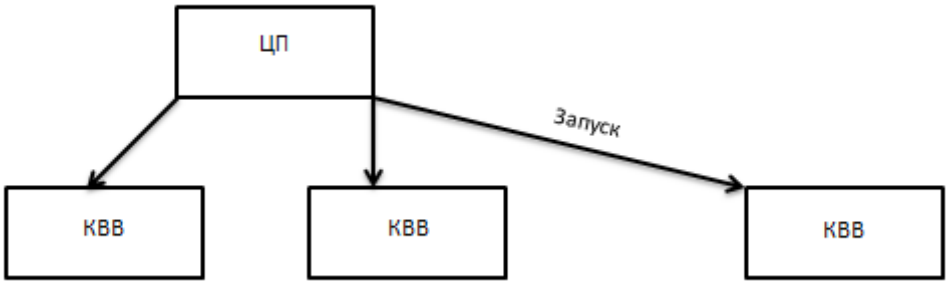


Рис. 9

Всі канали вводу / виводу працюють паралельно. ЦП займається виконанням прикладної програми мультипрограмування.

КВВ - канал вводу / виводу.

3)

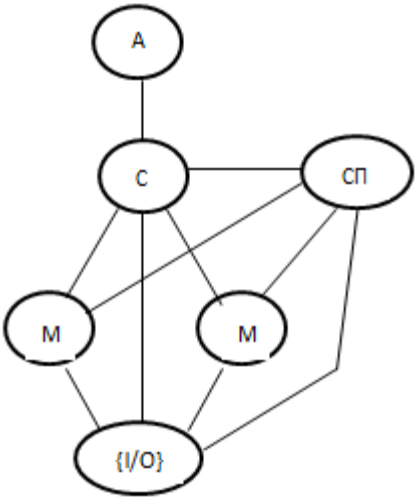


Рис. 10 Структурна організація ЕОМ 3-ого покоління

У рамках 3 структурного покоління були вирішені питання мультипрограмування, тобто були отримані необхідні і достатні умови для поєднання в часі циклів виконання прикладних програм і програм вводу / виводу. Тим самим третє структурне покоління фактично виключило подальшу можливість підвищення продуктивності машин у рамках мультипрограмної роботи. З'явилася необхідність у вирішенні завдань мультиобробки, тобто в збільшенні кількості структурних елементів у системі.

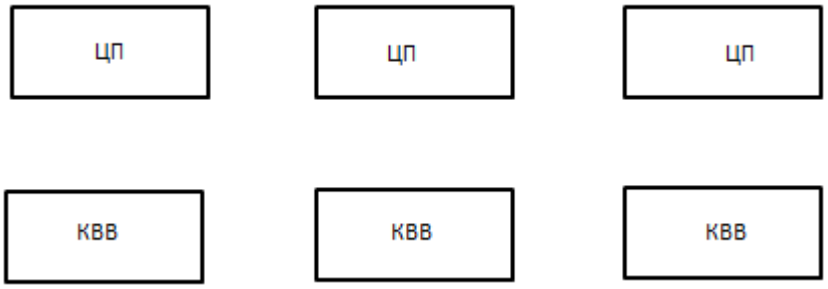
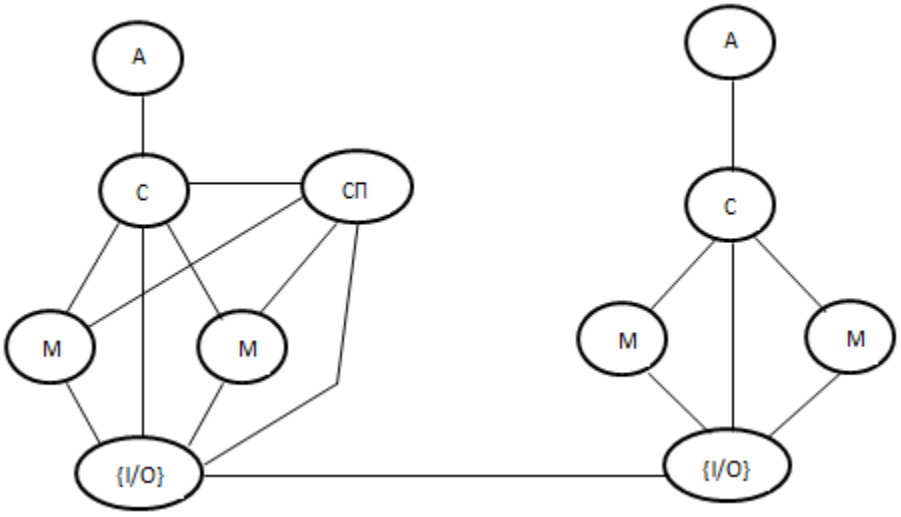


Рис. 11

4) Покоління зв’язних машин



Або

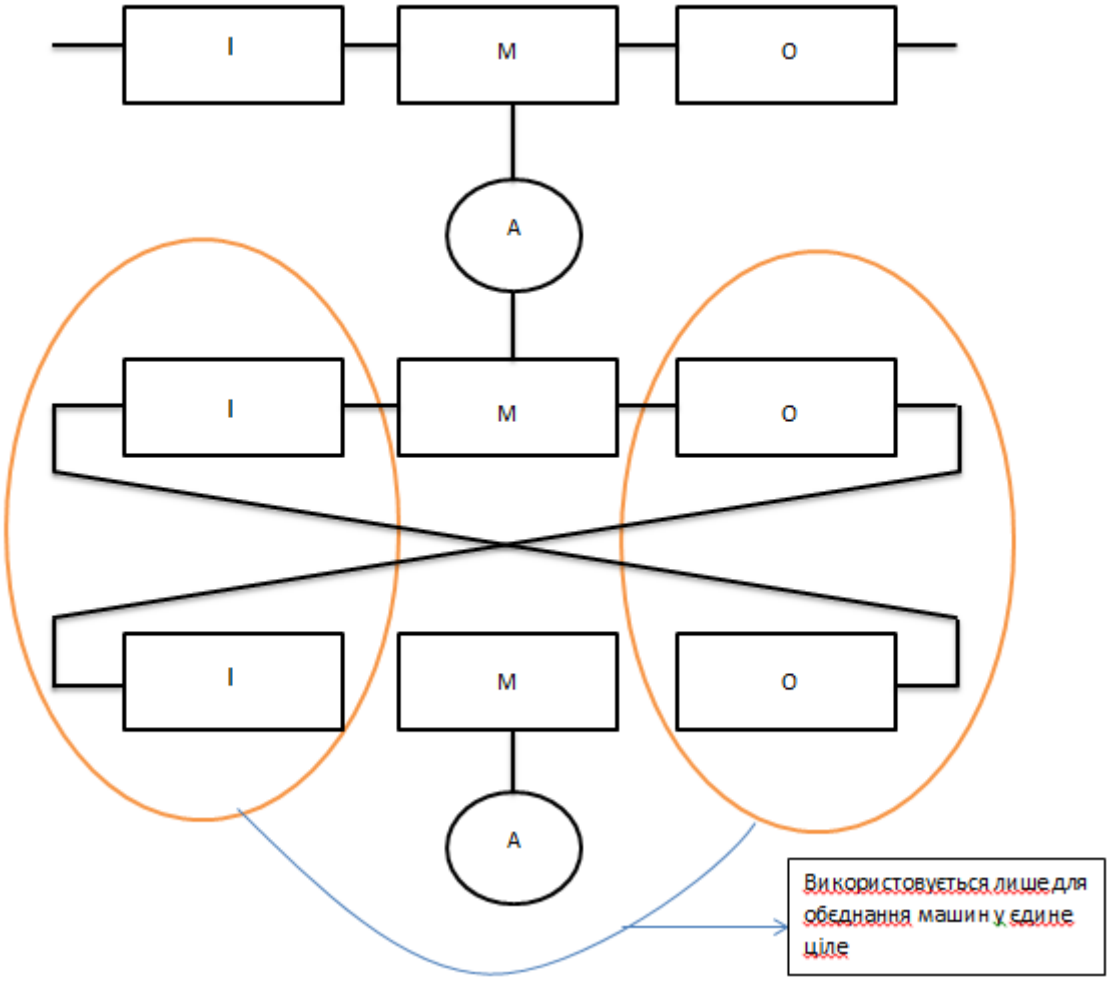


Рис. 12 Структурна організація ЕОМ 4-ого покоління

Недоліки:

1) Значні складності при перерозподілі інформації між машинами, що визначило можливість взаємодії лише на рівні загальних файлів, тобто фактично питання розпаралелювання вирішувалося на рівні завдань.

Завдання - користувальницька одиниця, що складається з кроків. (Задача - коли виділяються ресурси).

2) Необхідність використання додаткових пристроїв введення-виведення для вирішення завдань взаємодії.

Переваги:

Одночасно виконання декількох програм, але бібліотека для машин загальна.

5) Структурне покоління

3 покоління машин

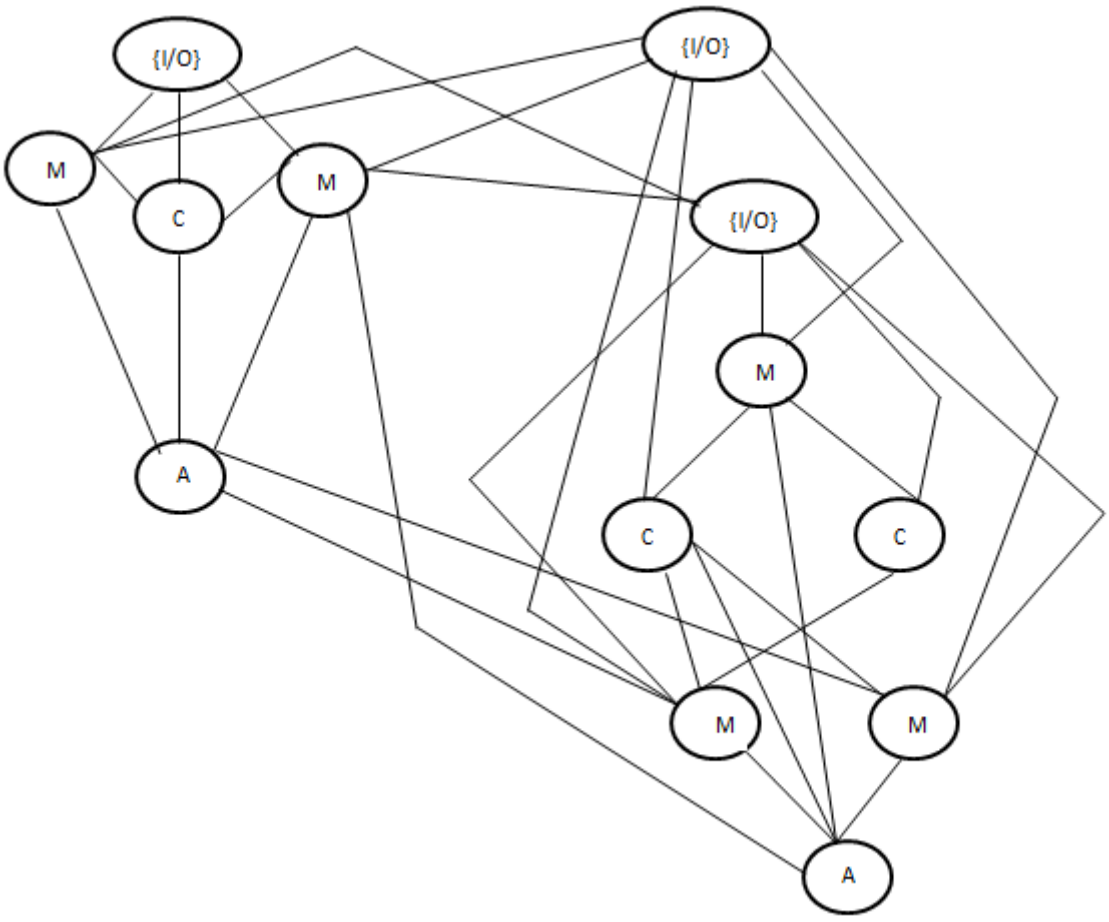


Рис. 13 Структурна організація ЕОМ 5-ого покоління

Кожний ЗП може бути пов'язаний зі своєю модифікованої пам'яттю. , Як і ЦП.

У 5 структурному покоління жорсткі фізичні зв'язки були замінені гнучкими логічними зв'язками. Дане завдання було вирішене на основі стандартних засобів сполучення, які надалі були названі інтерфейсами.

Інтерфейс - це уніфікована система зв'язків сигналів і алгоритмів.

Були реалізовані розвинені системи переривань або системи мультипрограмування і поділу часу, реалізовані природні і вимушені переривання.

Природні - переривання по вводу-виводу.

Вимушені - передача управління.

У рамках 3 структурного покоління система переривання виконувала тільки функції збору, обліку і аналізу стану структурних елементів системи.

У розвинених системах переривань одночасно з природним з'явилися вимушені переривання, а також з'явилися пріоритети різних програм і засобів ОС, які визначали дисципліну обслуговування всіх цих програмних елементів.

Машини п'ятого структурного покоління виконувалися на ІС малої і середньої складності.

4-е покоління характеризується:

- Реалізацією мультиобробки.

- Проблемною орієнтацією з якою будемо зв'язувати системи які в принципі можуть вирішувати будь-який клас задач, але з різною ефективністю, тобто ці системи виділяються деяким класом задач, які реалізуються максимально ефективно.

- Орієнтований на рішення матричних задач.

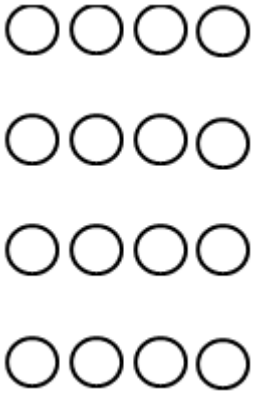


Рис. 14

Елементна база - БІСи ССІС.

5-е покоління:

Це покоління систем в яких реалізується масове розпаралелювання з використанням масштабованих систем.

Масштабовані системи - це такі системи, в яких кількість процесорних елементів може збільшуватись практично необмежено при збереженні топологічних властивостей системи.

Цілі паралелізму:

1) Підвищення надійності.

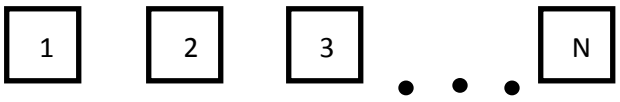
2) Підвищення відмовостійкості (живучості-можливості коректної роботи системи в умовах відмови).

3) Зменшення щільності потоків інформації по каналах зв'язку (зменшення вартості каналів зв'язку).

4) Підвищення продуктивності.

$$P_н = P_п, \qquad P_н - \text{Р номінальне}, \qquad P_п - \text{Р пікове}$$

Це сумарна продуктивність усіх процесорних елементів системи.



$$P_н = \sum_{i=1}^{N} S_i = N * S$$

$$P_р \leq P_н, \qquad P_р - \text{Р реальне}$$

$$E_{рс} = \frac{P_{рс}}{P_H}$$

$$E_{рн} = \frac{P_{рп}}{P_H}$$

$E_{рс}$ — реальна продуктивність системна(отримується при рішенні декількох задач)

$E_{рн}$
— реальна продуктивність користувача (всі процесори віддали одному користувачеві).

Способи організації паралелізму

Будемо розрізняти 3 способи:

1) Асинхронний

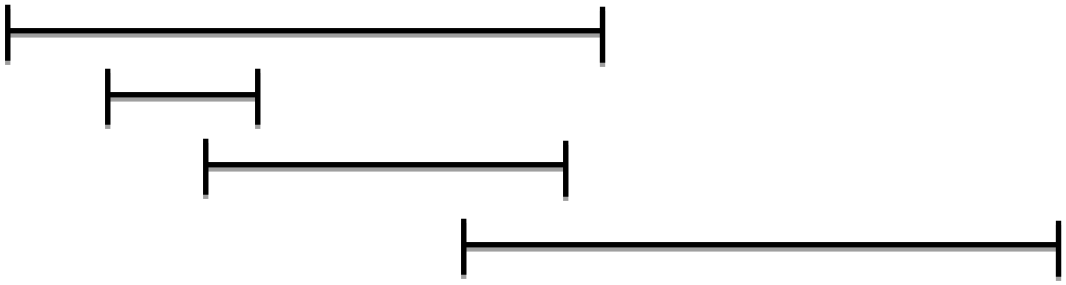
2) Синхронний

3) перекриває (конфлюентний).

Асинхронний спосіб зазвичай реалізується на рівні процесів або потоків.

Трансп’ютер - спеціальний елемент для побудови великих систем.

Процеси або потоки виконуються асинхронно і незалежно один від одного, тобто моменти початку, періоди розвитку і моменти завершення процесів або потоків ніяк не пов'язані між собою, можуть реалізуватися в будь-які моменти часу.



Прикладом асинхронної системи є мультипроцесорні системи (системи працюють із загальною пам'яттю).

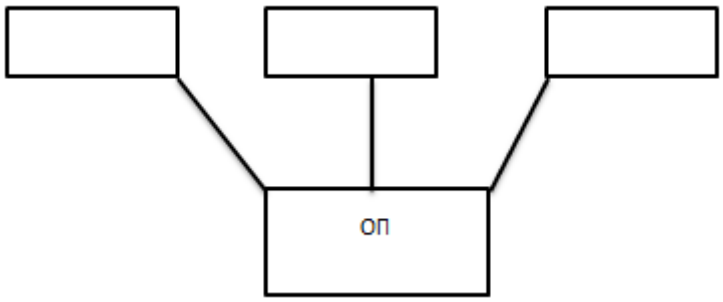


Рис. 15 Система зі спільною пам’яттю

У **синхронних системах** моменти початку та завершення елементів розпаралелювання збігаються між собою. В якості таких елементів можуть бути використані команди або операції.

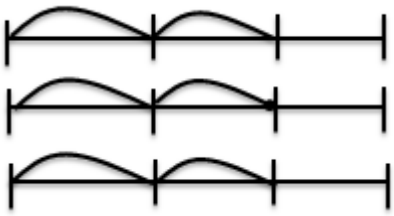


Рис. 16 Такти в синхронних системах

Тут питання взаємодії вирішуються значно простіше і будь-яка команда може реалізовувати питання взаємодії. Прикладом синхронної системи є матрична система. У цьому випадку у всіх ПЕ матриці буде виконаються одна і та ж операція.

Асинхронний і синхронний способи реалізації паралелізму - це суто просторове розпаралелювання.

Спосіб перекриття - це просторово тимчасовий спосіб розпаралелювання, за допомогою якого в максимальній мірі можна використовувати як частотні властивості елементної бази, так і вирішення питань розпаралелювання.

Тут питання розпаралелювання вирішуються на рівні операцій, але і на більш високому рівні. Прикладом є конвеєрна організація обчислень.

Рівні розпаралелювання

Будемо розрізняти наступні рівні:

Локальний паралелізм

1. Рівень завдань ($\{A\}, \{SD\}$) (не представляє інтересу)
2. Рівень задач(потоків) ($\{P\}, \{FD\}$)
3. Рівень команд ($\{O\}, \{D\}$)

Глобальний паралелізм

4. Мікроком ($\{\mu O\}, \{\mu D\}$)
5. Наноком ($\{\eta O\}, \{\eta D\}$)

Рівень задач

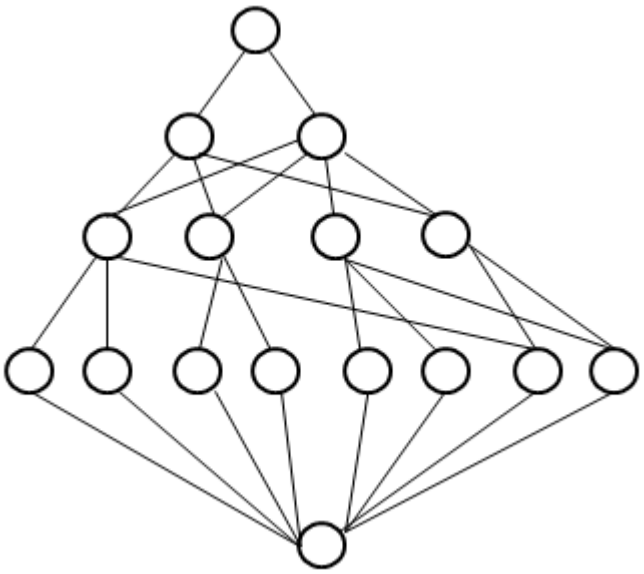


Рис. 17

Розпаралелювання на рівні завдань (потоків) часто називають розпаралелюванням незалежних гілок. Цей рівень розпаралелювання є найбільш використовуваним і створює гарні передумови для вирішення завдань розпаралелювання.

Розпаралелювання на рівні команд часто називають паралелізмом суміжних команд. Даний спосіб розпаралелювання, як правило, при вирішенні паралельних завдань виявляється не ефективним. Це пов'язано з тим, що суміжні команди, як правило, пов'язані між собою з даними чи з пам'яттю.

При такому зв'язку команди можуть виконаються тільки послідовно, однак існує спеціальний клас систем, у яких управління здійснюється не потоком команд, а потоком даних.

У цих системах використовуються спеціальні механізми для виділення всіх команд, для виконання яких готові всі вихідні дані, тобто виконання команд здійснюється не в природній послідовності, а в міру готовності вихідних даних для виконання команд.

Управління потоком даних Data Flow

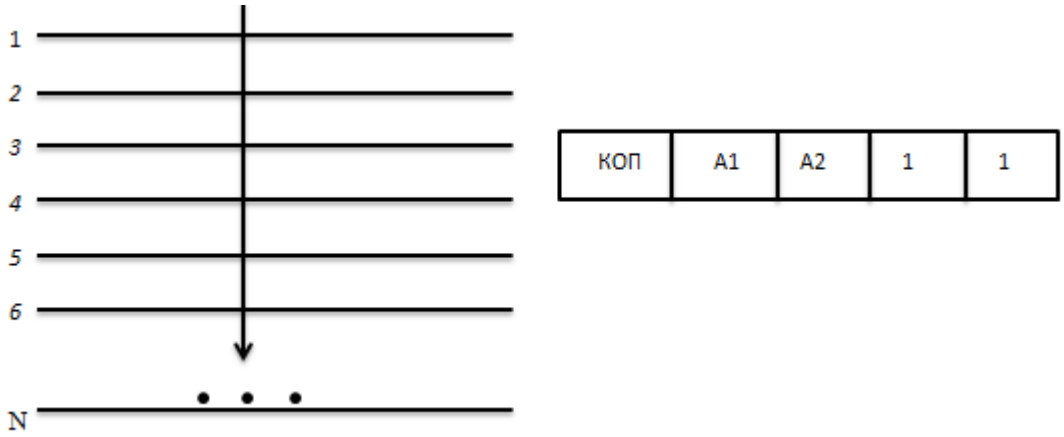
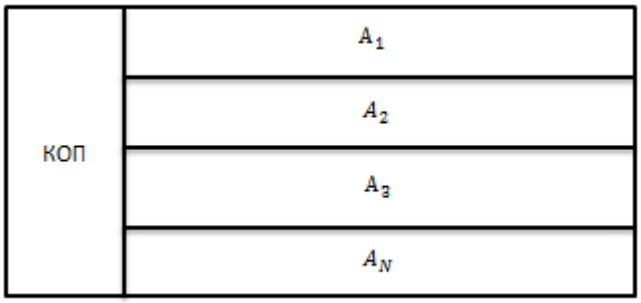


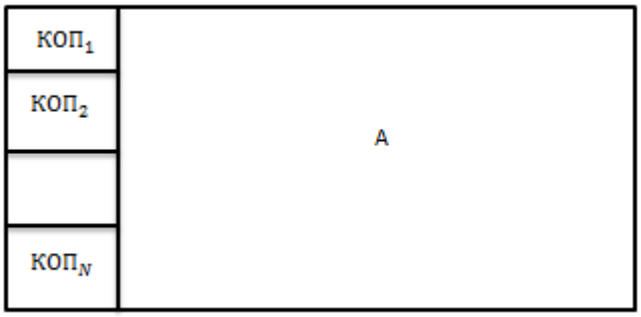
Рис. 18

У рамках глобальних рівнів розпаралелювання будемо розрізняти природний, приватний і загальний паралелізм

Природний



Приватний

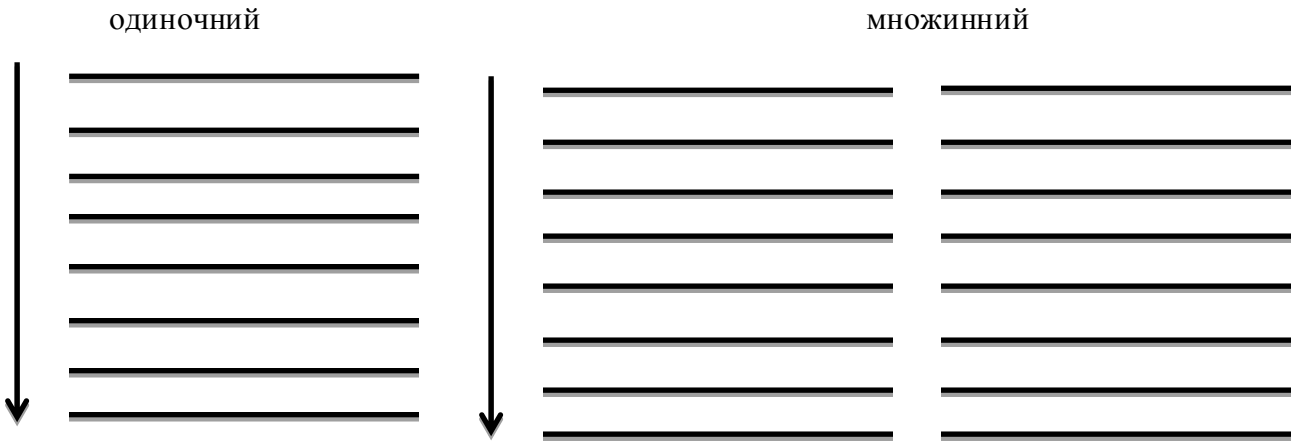


Загальний

КОП$_1$	A_1
КОП$_2$	A_2
КОП$_N$	A_N

Класифікація ОС з точки зору паралелізму

Класифікація Флінна відповідно до якої замість терміна «паралелізм» вноситься термін «множинність», точніше одиночний або множинний потоки команд / даних.



ОК- одиночний потік команд (SISD).

OD – //- дані(SISD).

OKMD (SIMD),MKOD (MISD),MKMD (MIMD).

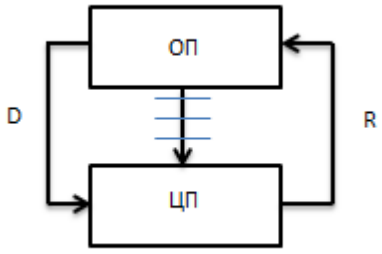


Рис. 19 OKOD

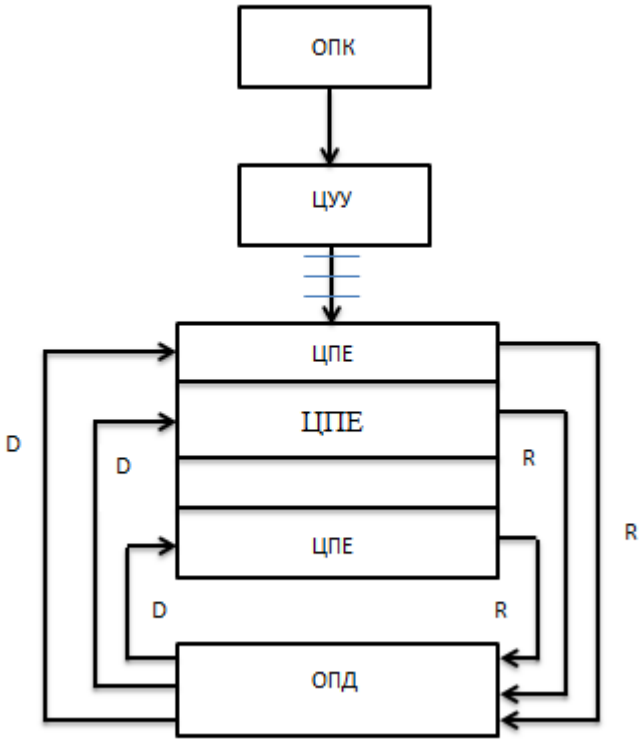


Рис. 20 OKMD

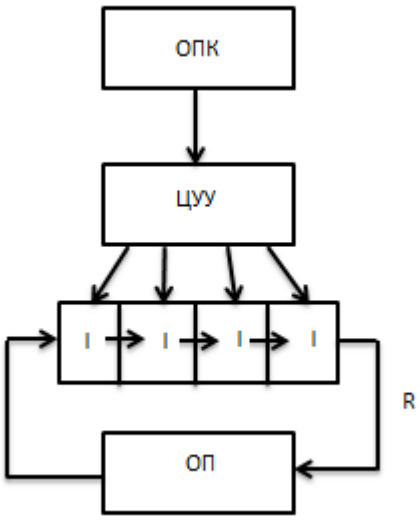


Рис. 21 MKDO

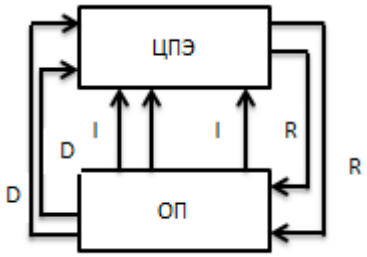


Рис. 22 MDMK

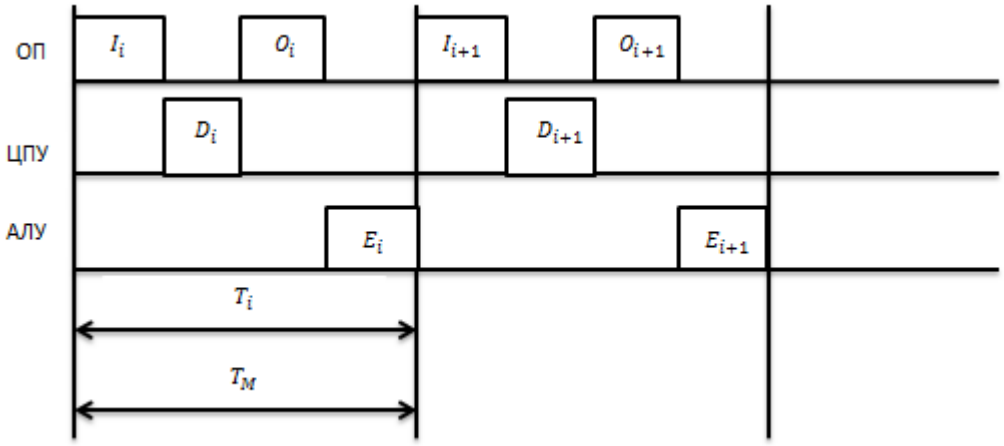


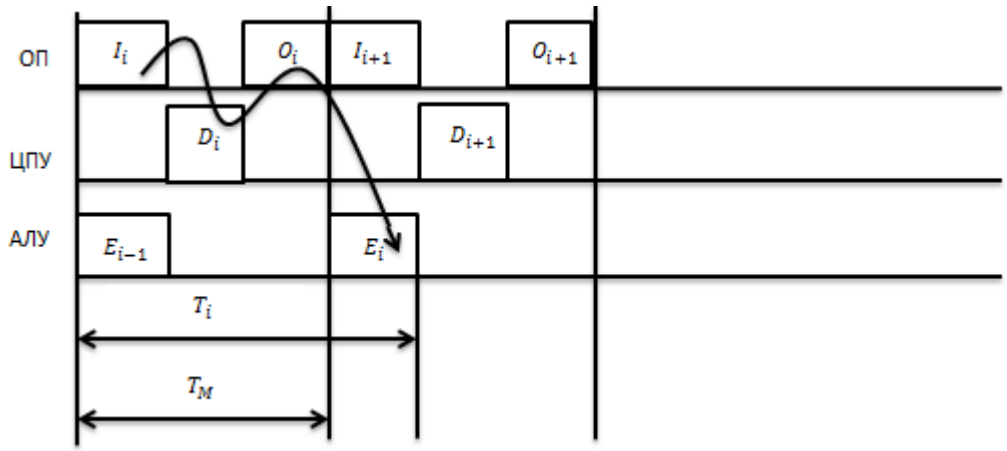
Рис. 23 OKOD (SISO)

$T_M = T_i$

Задача:

$T_M \ll T_i$

Варіанти:



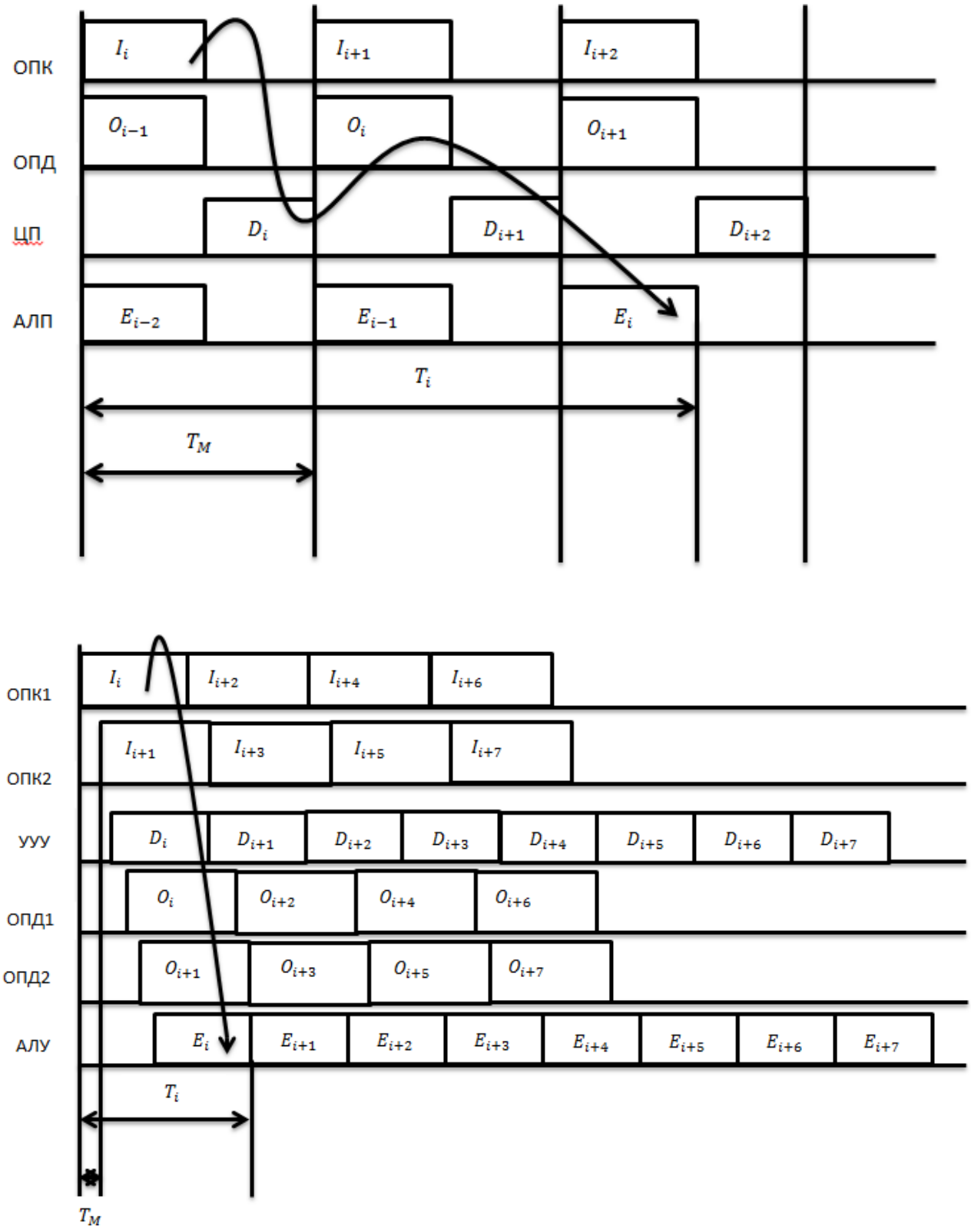


Рис. 24 Колективна пам'ять даних і пам'ять команд

$T_M \ll T_i$

Спроби подальшого збільшення продуктивності машин класу SISD призводять до необхідності збільшення або модулів ПУ, або модулів АЛУ, а це відповідно означає перехід до систем класу MISD або SIMD.

Таким чином час роботи дешифратора команд, визначає мінімальний цикл машини.

SIMD. В рамках даного класу будемо розрізняти наступну паралельну організацію:

1) матрична;

2) конвеєрна (векторна);

3) асоціативна;

Матрична організація

Розрізняють матричні організації з індивідуальною пам'яттю і з загальною (роздільною) пам'яттю.

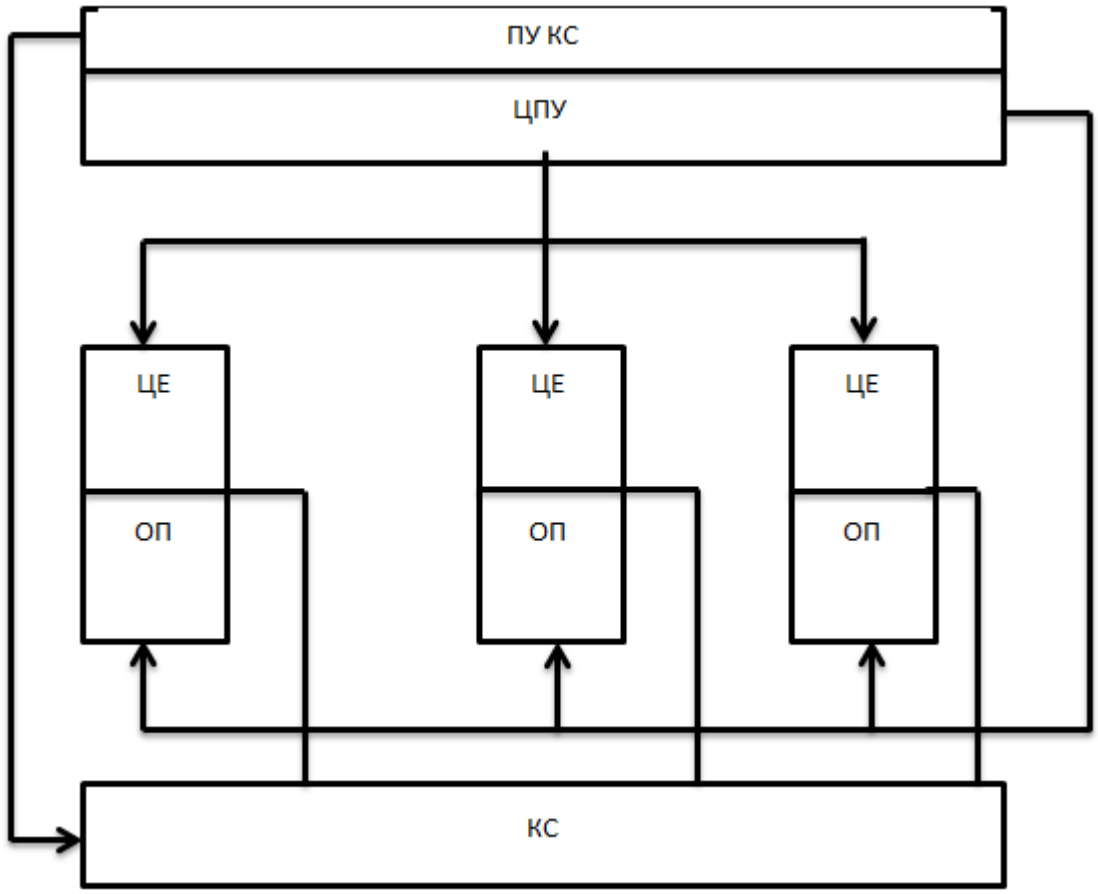


Рис. 25 Структурна схема КС матричної організації із індивідуальною пам’яттю

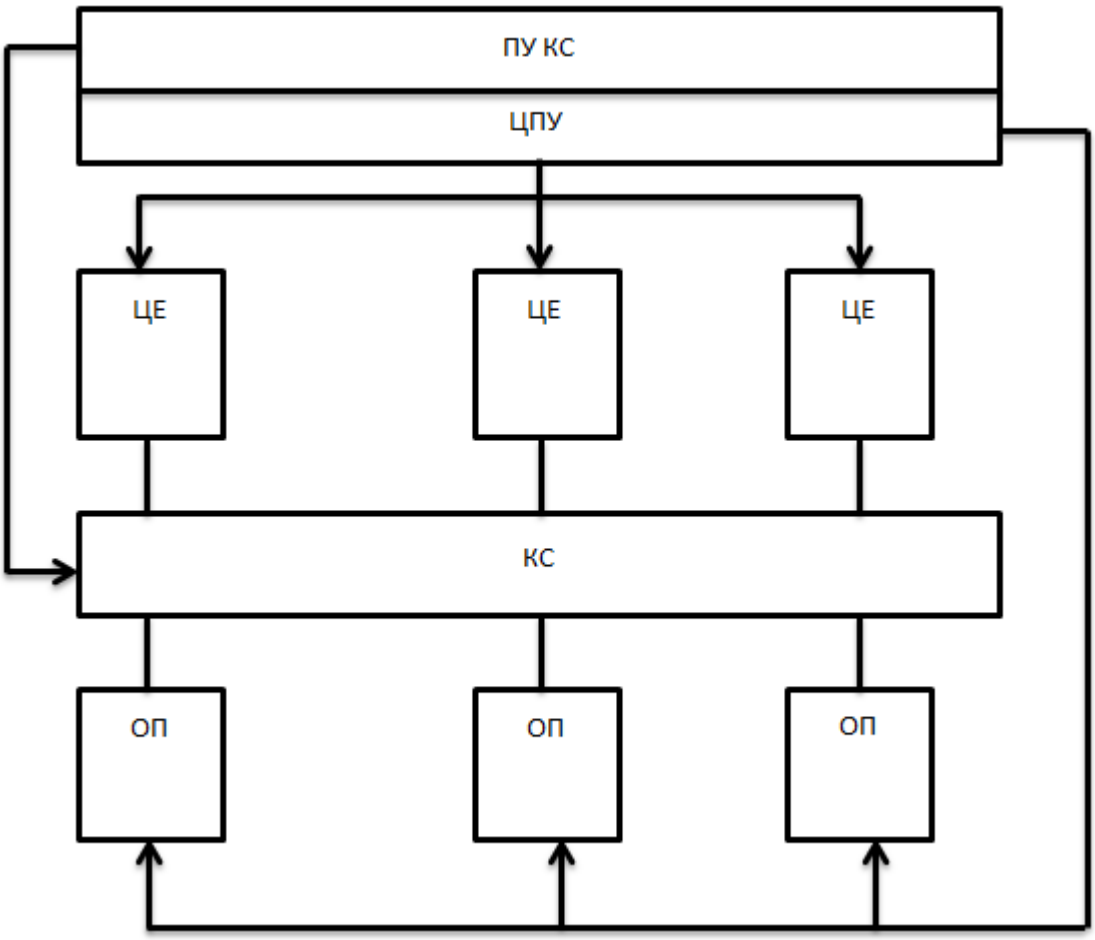


Рис. 26 Структурна схема КС матричної організації із роздільною пам’яттю

Особливості:

1) Машини з матричними системами - це за своєю суттю звичайні машини 1 і 2 поколінь з тією лише відмінністю, що замість звичайного ЦП тут використовується матричний ЦП, всі елементи якого виконують одну і ту ж команду ЦПУ.

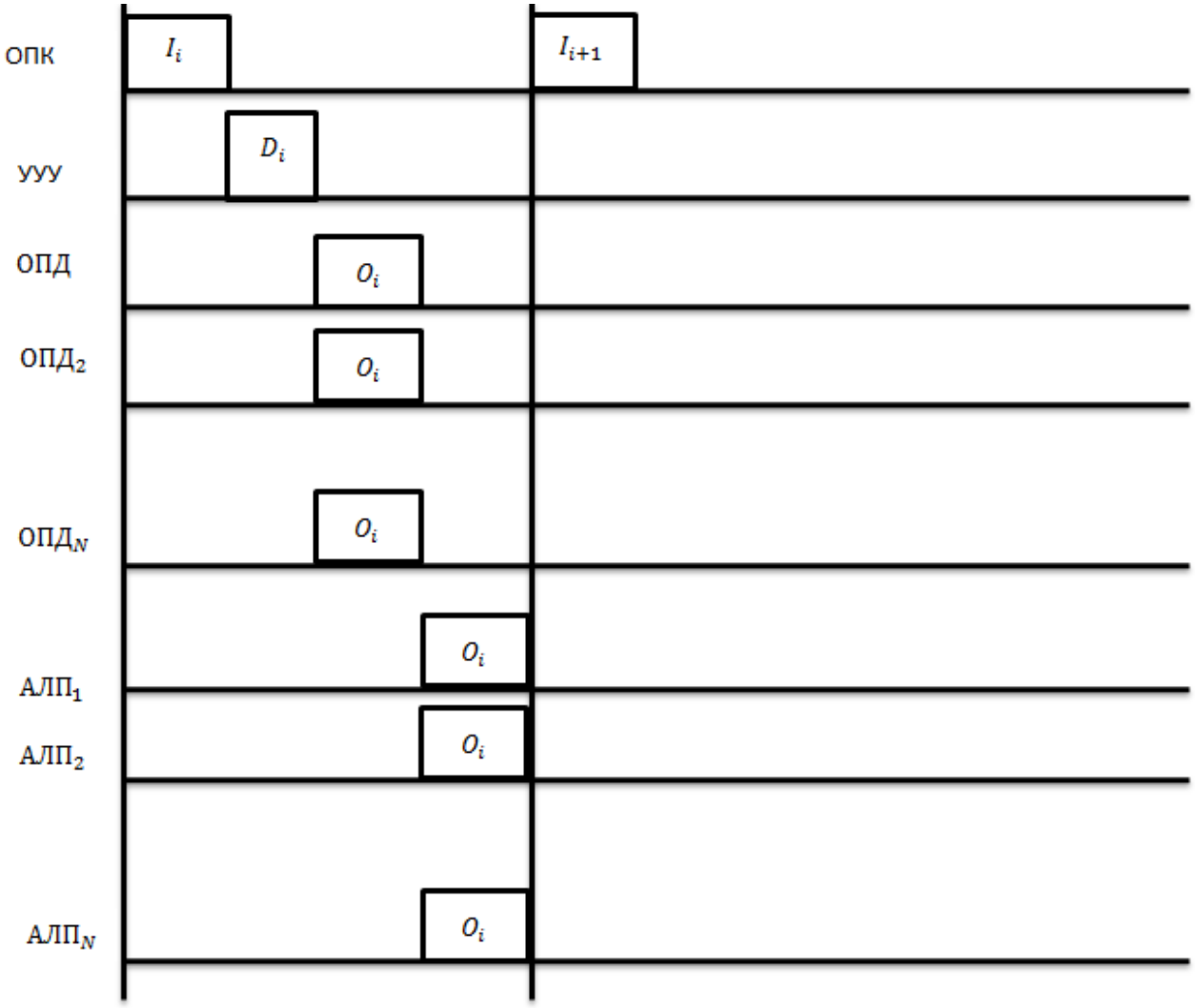


Рис. 27 Приклад роботи матричної КС

2) Машини класу SIMD орієнтовані на широке розпаралелювання оскільки всі файли управління як і в машинах 1 і 2 покоління зосереджені в єдиному центральному ПУ то це призводить до використання досить простих і дешевих ЦПЕ, що в свою чергу дозволяє реалізовувати в одній машині широке розпаралелювання (за своєю суттю ЦПЕ це АЛП). Слід так само враховувати що тут має місце в один і той же момент часу тільки один процес, що у свою чергу визначає найпростішу організацію даної системи.

3) Враховуючи можливість широкого розпаралелювання в цих системах тобто використання великої кількості ЦПЕ в цих системах, використання загальної (роздільної) ОП тут недоцільно так як при цьому буде створюватися величезна кількість конфліктних ситуацій, а отже великі черги на використання ОП. Через це другий варіант вважається неприйнятним і широке застосування отримав перший варіант.

Першими матричними системами були такі система як система Унгера, Solomon, Illiac - 4.

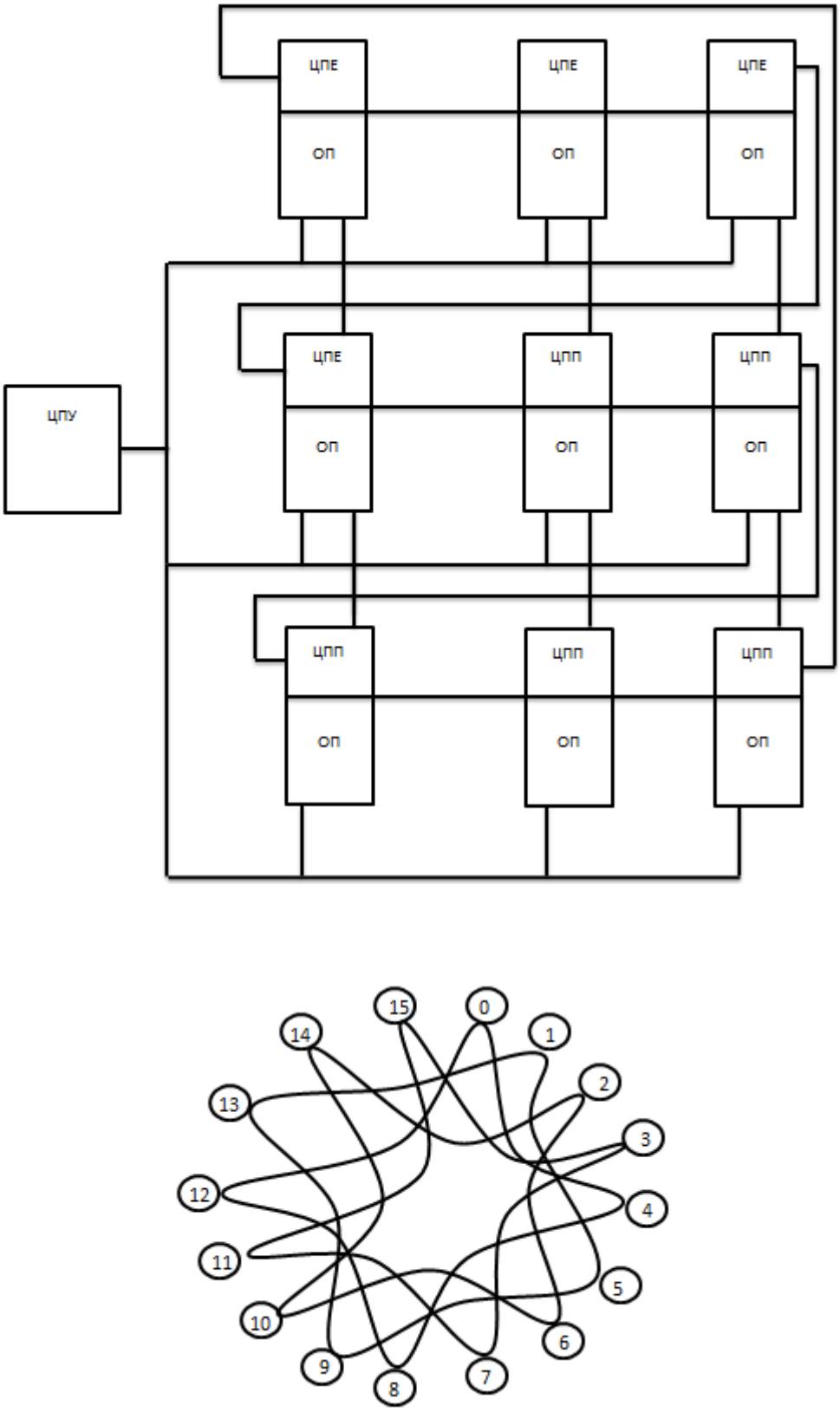


Рис. 28 КС і хордова система зв’язку

$\sqrt{N}$ -1 – максимальна кількість кроків.

Система Унгера 1024 32 * 32 - Не було механізму маскування що істотно ускладнювало вирішення питань програмування при необхідності виконання обробки рядків (стовпців) і особливо скалярних операцій.

Система Унгера розроблялася як спеціалізована система для обробки спец. образів.

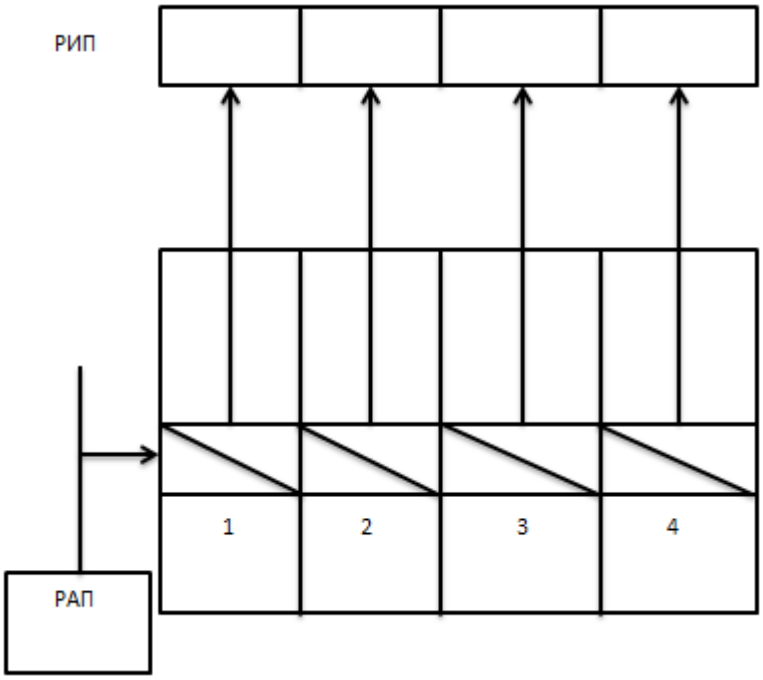


Рис. 29

В системі Solomon був введений механізм маскування. Регістр маски - в ЦПУ і в кожному ЦПЕ організація пам'яті така ж.

У Illiac - 4 доданий 64х розрядний індексний регістр, який визначав базу.

Ап.а.=B+Ai;

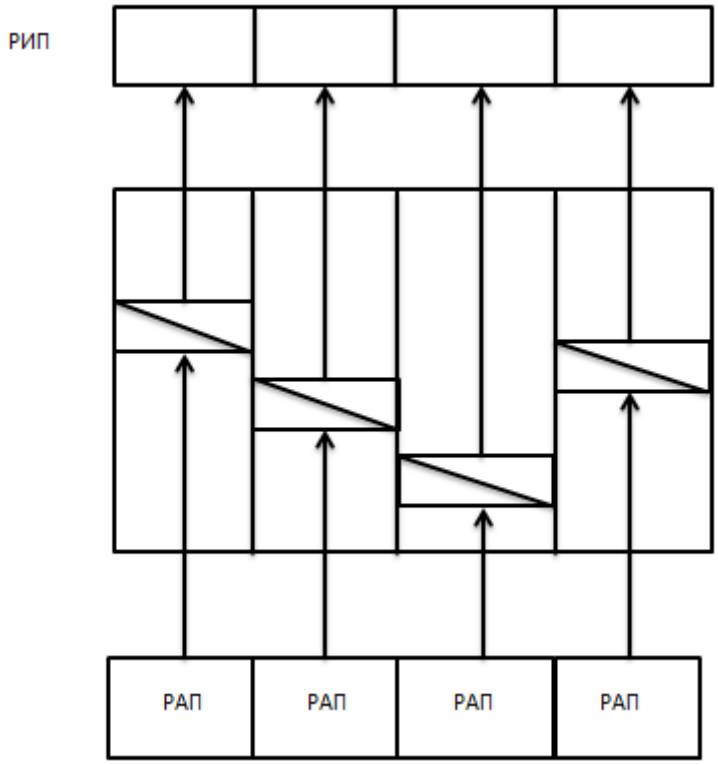


Рис. 30

У Illiac - 4 розроблений механізм маскування і механізм рішення щодо незалежної адресації. Крім того в якості ЦПЕ тут використовувався потужний 64х розрядний ПЕ.

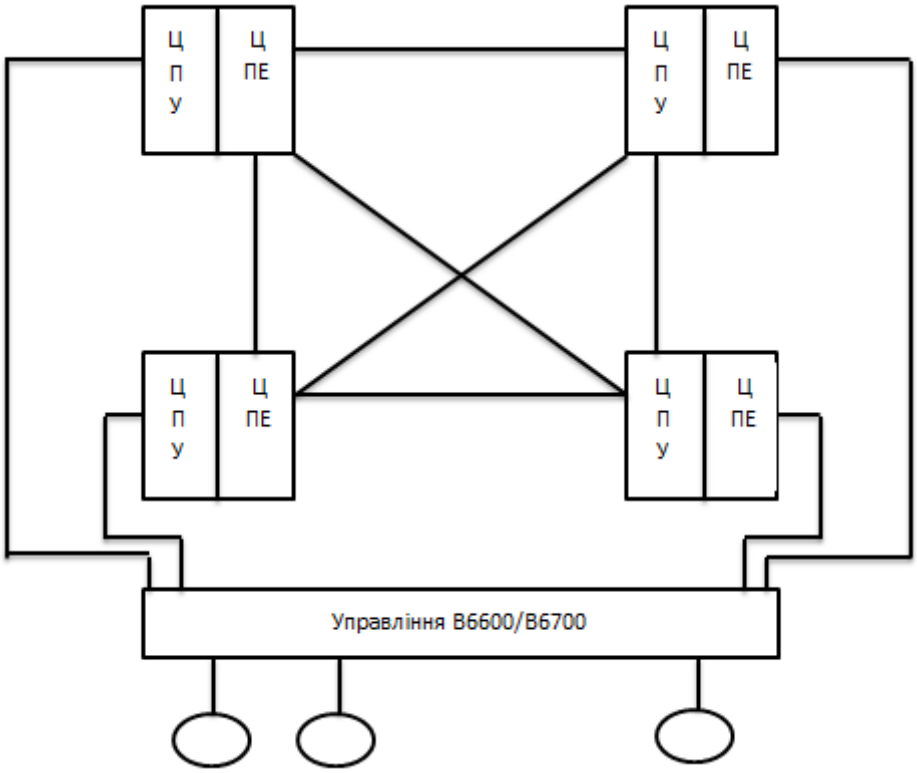


Рис. 31

Закон Грома

Продуктивність пропорційна квадрату вартості $P \equiv C^2$

$P = 10^9\ flops$

$P = 4 * 10^6\ flops\ (1\ ПЭ) * 256 = 10^9$

Були створені зовсім нові елементи пам'яті для зчитування лазерним променем з ємністю 10^{12}.

Були розроблені нові диски ємністю 10^9.Крім того, повинна була бути створена ОП на тонких плівках. Елементна база - ІС середньої складності повинні були бути розроблені.

ОП на тонких плівках і елементна база володіли низьким коефіцієнтом виходу (втрата продуктивності на 20%).

Перша машина з напівпровідниковою пам'яттю - Illiac - 4.

В результаті залишили тільки один квадрант.

Продуктивність = $2 * 10^6$ замість 10^9.

Недоліки матричних процесорів:

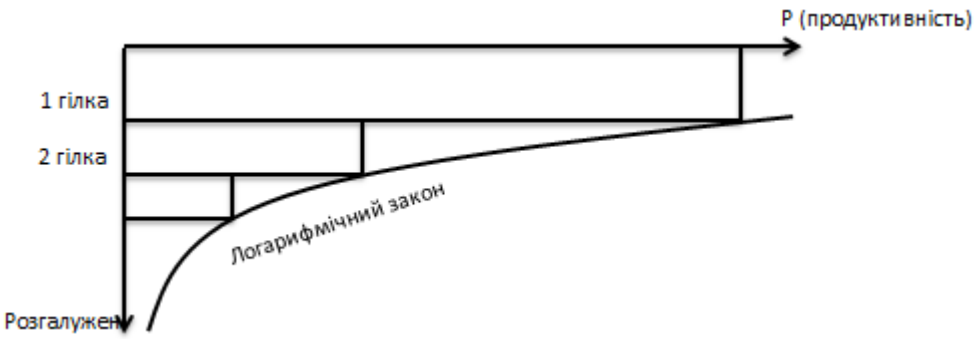
1) Невідповідність логічного та фізичного вектора.

Логічний - (Наше завдання потрібно всього 8 або 2 * 2 або 4 * 4).

Фізичний - (3 * 8 = 64 відс.).

2) Значні витрати часу на виконання операції взаємодії (багато пересилань, 1 кім. = 1 пересилання).

3) Деградація продуктивності при великій кількості гілок.



На початковому етапі ми припускаємо що є відповідність логічного та фізичного векторів, тобто працюють всі ПЕ і продуктивність max.

З різних ПЕ різні дані і в одних умовах розгалуження виконується, в інших немає.

Але у нас пристрій SIMD і ми можемо йти тільки по одному шляху. Це означає що половина ПЕ працювати не буде (будуть в стані очікування).

Векторні конвеєрні системи

На кожному робочому місці виконується одна і та ж операція (на відміну від конвеєра).

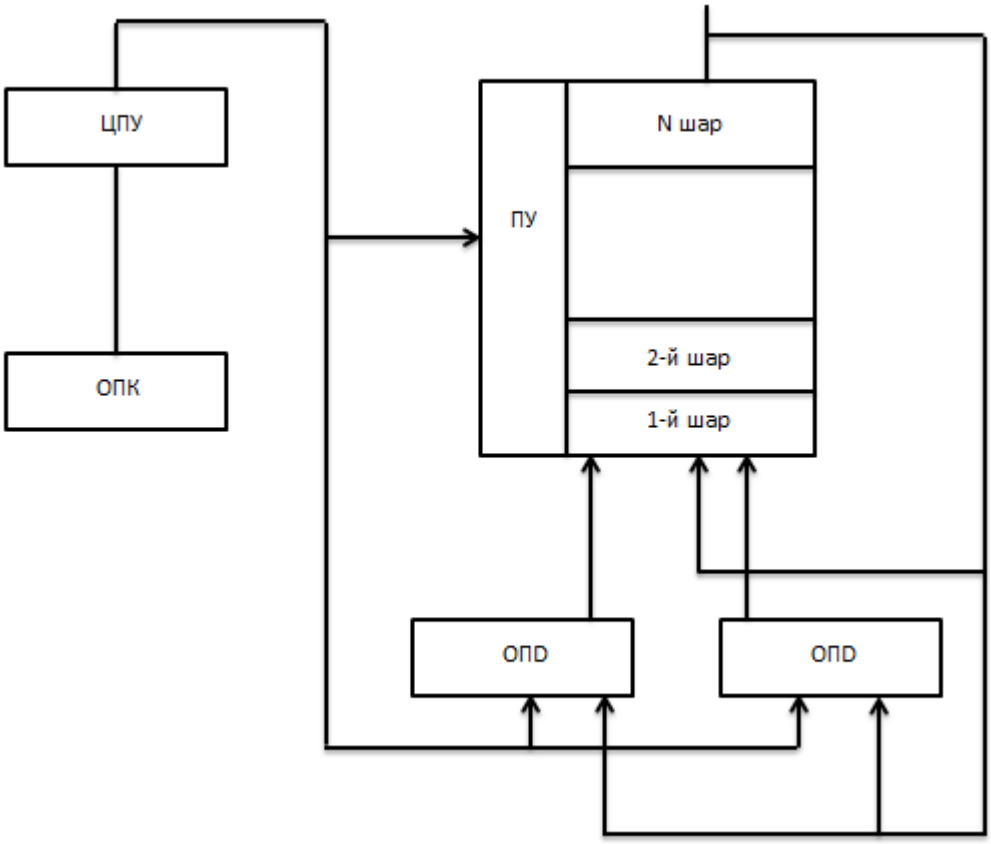


Рис. 32 Структурна схема векторної конвеєрної системи

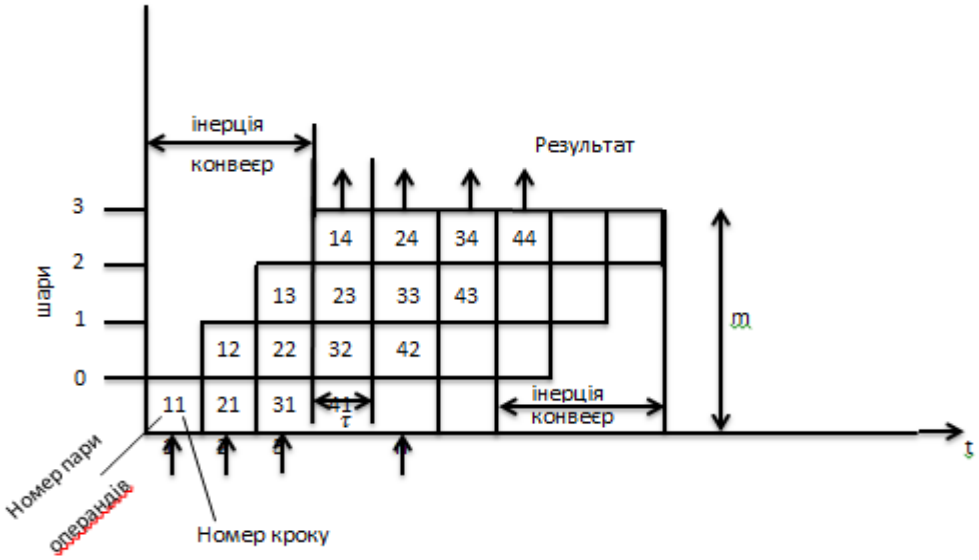


Рис. 33 Робота векторної конвеєрної системи

$$\tau \to min$$

$$T_n = n * \ \tau + (m-1) * \ \tau = (n+m-1) * \tau$$

$$T_i = \frac{(n+m-1) * \tau}{n}$$

$$P = \frac{1}{T_i} = \frac{n}{(n+m-1)*\tau} = \frac{1}{\left(1+\frac{m-1}{n}\right)*\tau}, \text{при } n \to \infty, = \frac{1}{\tau}$$

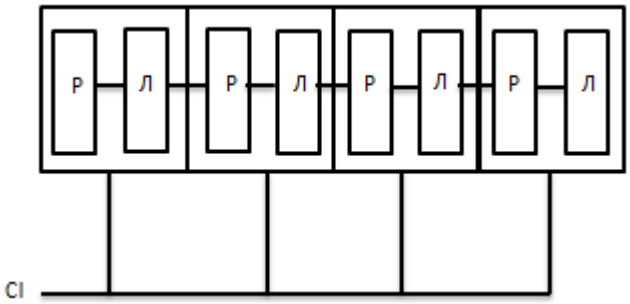


Рис. 34 Конвеєр pipeline

Р – регістр Л – логіка

Темп визначається найбільш повільним шаром

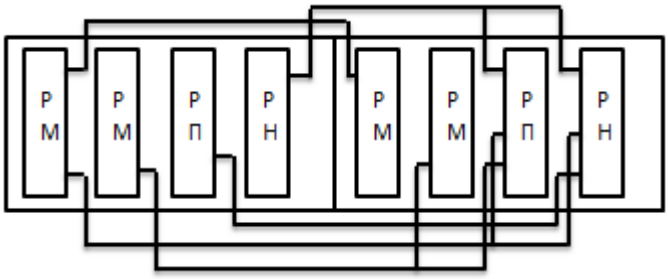


Рис. 35

РМ - регістр розмножувальний

РП - регістр пам'яті

РН - регістр накопичувач

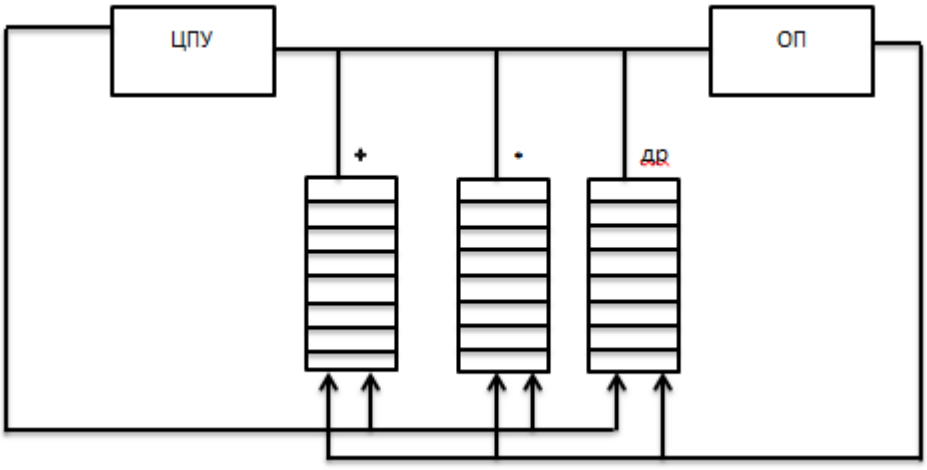


Рис. 36 1 варіант

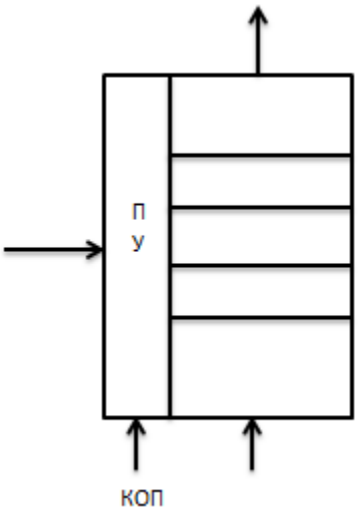


Рис. 37 2 варіант

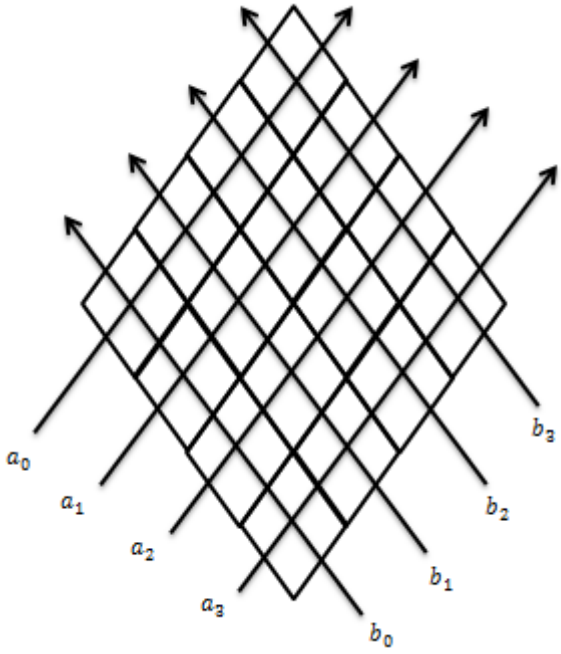
Систолічні процесори

Систолічні процесори - це деякий розвиток конвеєрних, спрямований на максимальне використання частотних можливостей елементної бази.

Досягається за рахунок:

1) Реалізації тільки близьких зв'язків, тобто реалізація принципу «близькодії».

2) Функції управління зануренням в структуру. Це досягається за рахунок введення структурних особливостей системи, а з іншого боку - на основі спеціальних алгоритмів обробки операцій.



$$\sum_{i-0}^{N-1} aibi$$

Рис. 38

Асоціативні процесори

Це ті ж матричні процесори у яких в якості ОП використовується асоціативна пам'ять. У асоціативної пам'яті зазвичай виконується обробка масиву, тобто при використанні асоціативної пам'яті природним чином вирішуються питання паралелізму. Крім того істотна частина частки обробки даних вирішується всередині асоціативної пам'яті, що істотно зменшує число операцій пересилки.

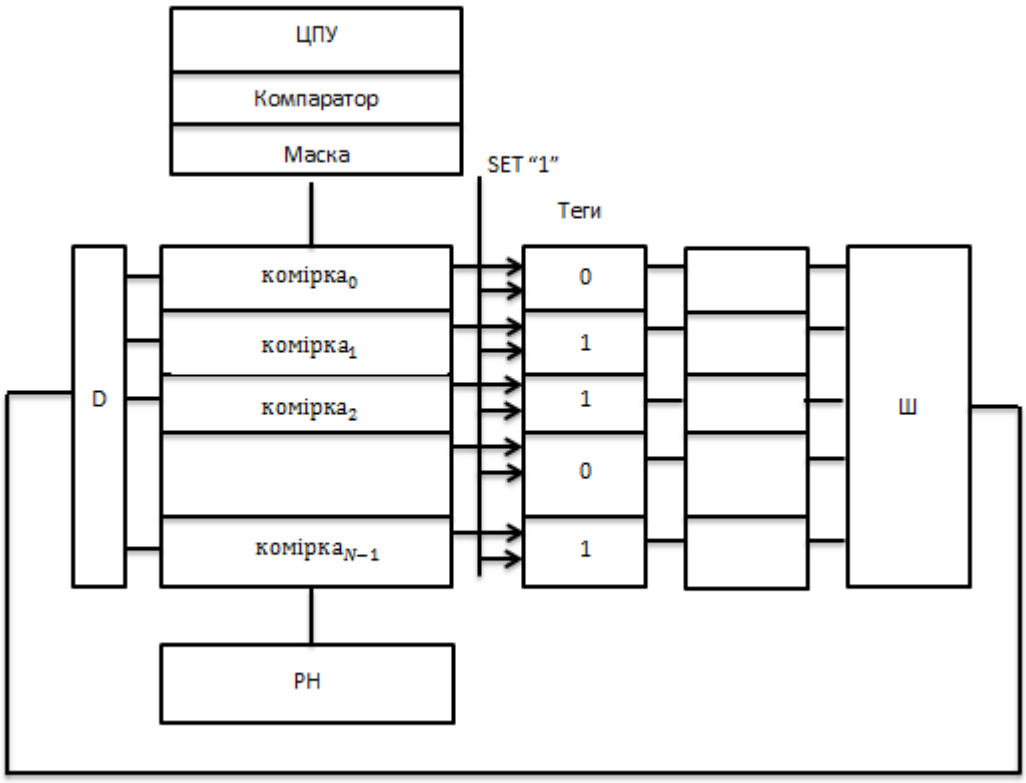


Рис. 39

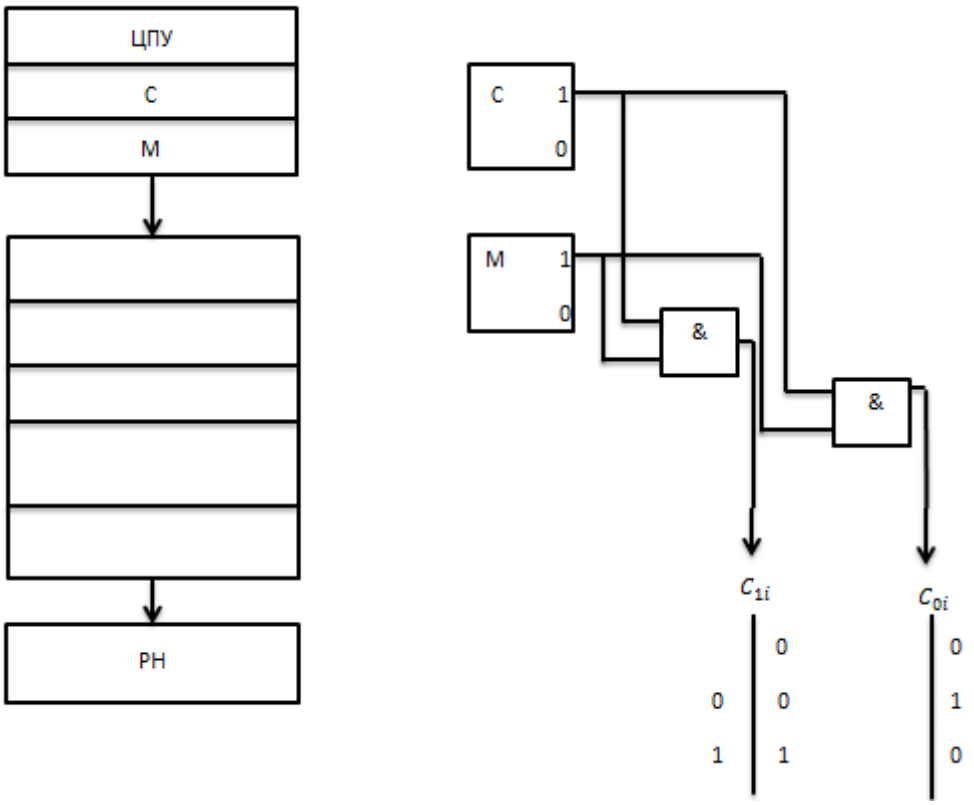


Рис. 40

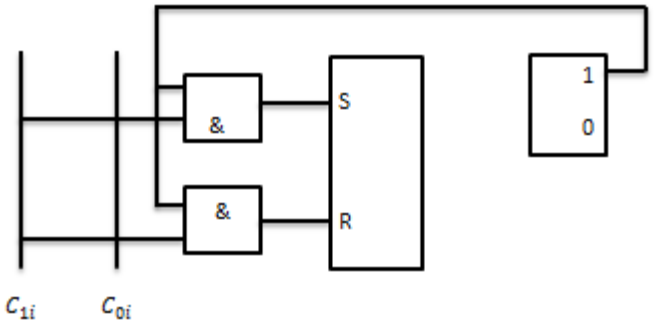


Рис. 41 Запис

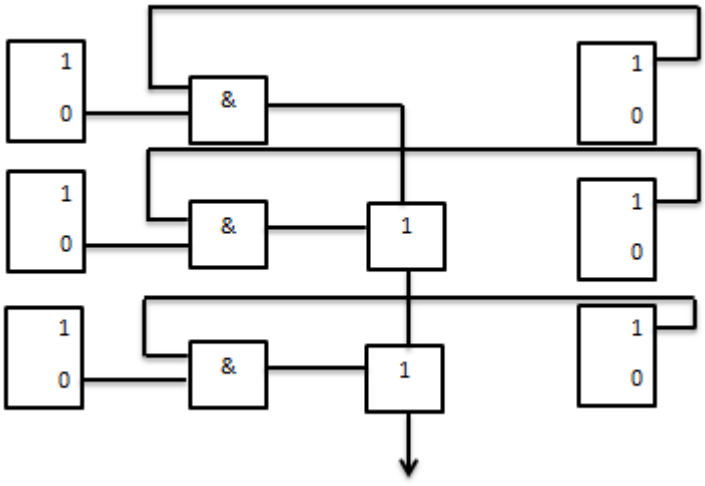


Рис. 42 Зчитування

Пошук максимального числа

C = 11..11 (макс. число)

M = 10..0 (маска)

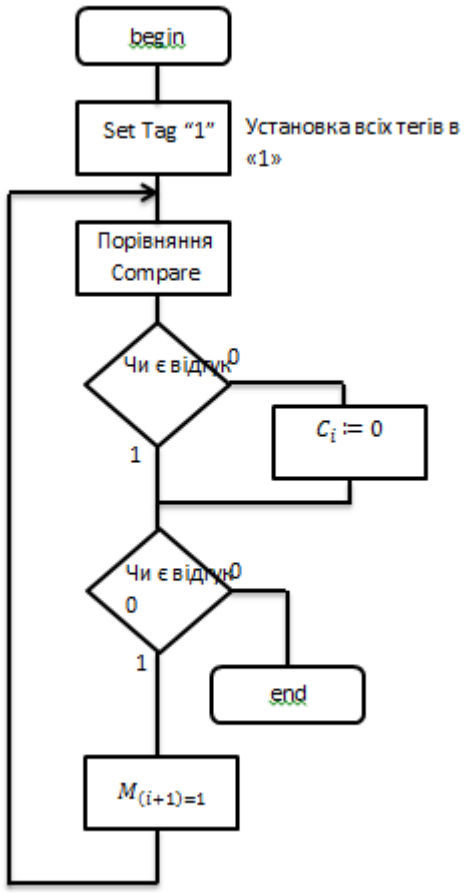


Рис. 43 Алгоритм пошуку максимального числа

АП з вибіркою паралельно за розрядами, паралельно за словами. Це досить складний вид асоціативної пам'яті, сьогоднішній день практично здійснений, тому його розглядати не будемо.

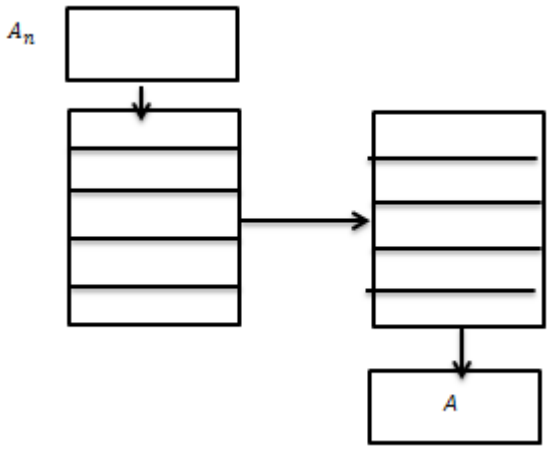


Рис. 44 A_n з вибіркою послідовно за розрядами, паралельно за словами.

Найбільш використовуваний варіант зображений на рис. 45.

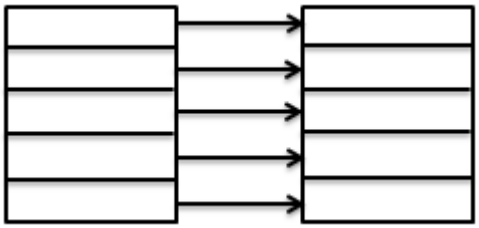


Рис. 45

АП зі спеціальною вибіркою, можна вибирати з різних слів різну кількість розрядів

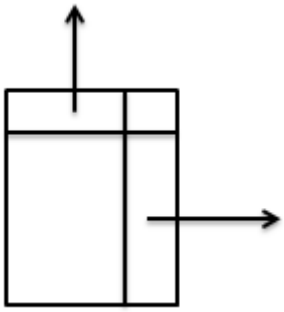
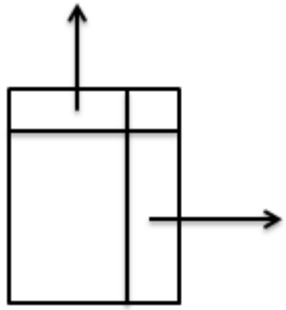


Рис. 46 Пам'ять ортогонального типу

Більш серйозне рішення можна вибирати по частинах. Наприклад машина STARAN (рис. 47).



STARAN:

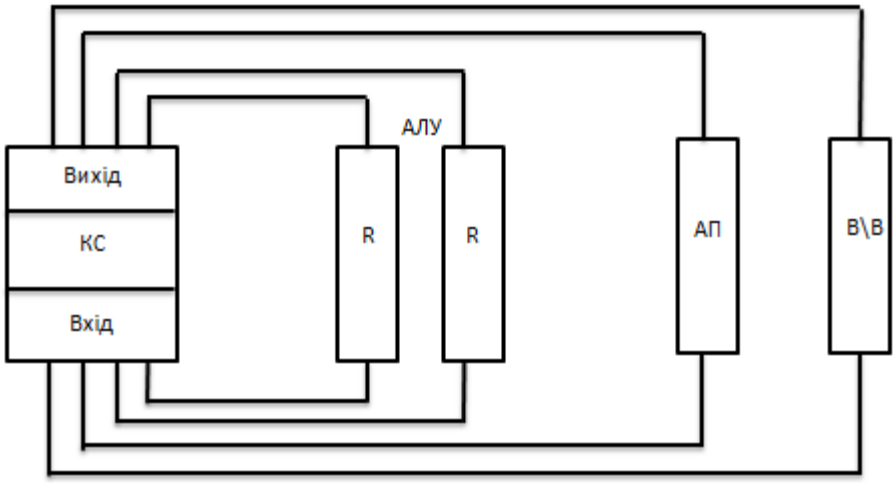
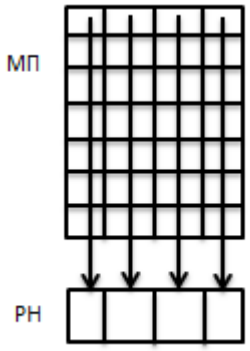


Рис. 47

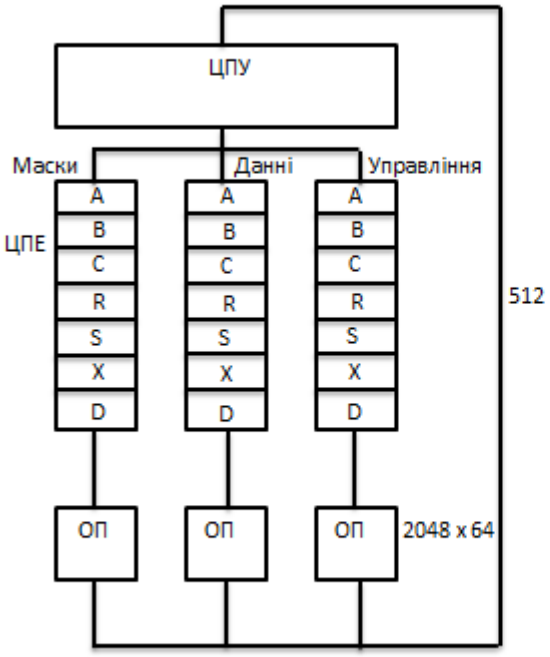
Можна зчитувати

1x24 ,2x128 ,4x64 ,8x32 ,256x1 – розрядні слова.



$$\begin{matrix} a_{00}a_{01}a_{02}a_{03} \\ a_{10}a_{11}a_{12}a_{13} \\ a_{20}a_{21}a_{22}a_{23} \\ a_{30}a_{31}a_{32}a_{33} \end{matrix} = \begin{matrix} a_{00}a_{01}a_{02}a_{03} \\ a_{13}a_{10}a_{11}a_{12} \\ a_{22}a_{23}a_{20}a_{21} \\ a_{31}a_{32}a_{33}a_{30} \end{matrix}$$

Приклад представлено на рис. 48.



A – регістр накопичувач

B – регістр переносу

C – регістр множника

R – регістр зв'язку

S – дод. регістр накопичувач

X – індексний регістр

D – регістр стану ,одним з видів якого є маска.

Рис. 48

ПЕ1
ПЕ2
ПЕ3
ПЕ4

$$\begin{matrix} x_{00}x_{01}x_{02}x_{03} \\ x_{10}x_{11}x_{12}x_{13} \\ x_{20}x_{21}x_{22}x_{23} \\ x_{30}x_{31}x_{32}x_{33} \end{matrix} \quad \begin{matrix} y_{00}y_{01}y_{02}y_{03} \\ y_{10}y_{11}y_{12}y_{13} \\ y_{20}y_{21}y_{22}y_{23} \\ y_{30}y_{31}y_{32}y_{33} \end{matrix}$$

ПЕ1	ПЕ2	ПЕ3	ПЕ4
$A = x_{00}y_{00}$	$A = x_{00}y_{11}$	$A = x_{00}y_{12}$	$A = x_{00}y_{13}$
$S = S + x_{01}y_{00}$	$S = S + x_{01}y_{11}$	$S = S + x_{01}y_{12}$	$S = S + x_{01}y_{13}$

$$\begin{matrix} x_{00}x_{01}x_{02}x_{03} \\ x_{10}x_{11}x_{12}x_{13} \\ x_{20}x_{21}x_{22}x_{23} \\ x_{30}x_{31}x_{32}x_{33} \end{matrix} * \begin{matrix} y_{00}y_{01}y_{02}y_{03} \\ y_{10}y_{11}y_{12}y_{13} \\ y_{20}y_{21}y_{22}y_{23} \\ y_{30}y_{31}y_{32}y_{33} \end{matrix}$$

$$x_{00} * y_{00} + x_{01} * y_{10} + x_{02} * y_{20} + x_{03} * y_{30} = Z_{00}$$

$$x_{00} * y_{01} + x_{01} * y_{11} + x_{02} * y_{21} + x_{03} * y_{31} = Z_{01}$$

$$x_{00} * y_{02} + x_{01} * y_{12} + x_{02} * y_{22} + x_{03} * y_{32} = Z_{02}$$

$$x_{00} * y_{03} + x_{01} * y_{13} + x_{02} * y_{23} + x_{30} * y_{33} = Z_{03}$$

ПЕ1	ПЕ2	ПЕ3	ПЕ4
$x_{00}y_{00}$	$x_{00}y_{01}$	$x_{00}y_{02}$	$x_{00}y_{03}$
$S = S + A$	$S = S + A$	$S = S + A$	$S = S + A$
$x_{01}y_{10}$	$x_{01}y_{11}$	$x_{01}y_{12}$	$x_{01}y_{13}$
$S = S + x_{01}y_{10}$	$S = S + x_{01}y_{11}$	$S = S + x_{00}y_{12}$	$S = S + x_{01}y_{13}$
$x_{02}y_{20}$	$x_{02}y_{21}$	$x_{02}y_{22}$	$x_{02}y_{23}$
$S = S + x_{02}y_{20}$	$S = S + x_{02}y_{21}$	$S = S + x_{02}y_{22}$	$S = S + x_{02}y_{23}$
$x_{03}y_{30}$	$x_{03}y_{31}$	$x_{03}y_{32}$	$x_{03}y_{33}$
$S = S + x_{03}y_{30}$	$S = S + x_{03}y_{31}$	$S = S + x_{03}y_{32}$	$S = S + x_{03}y_{33}$

MISD (MKOD)

Це той же конвеєр що і векторний процесор, але на кожному шарі будуть виконуватися різні операції.

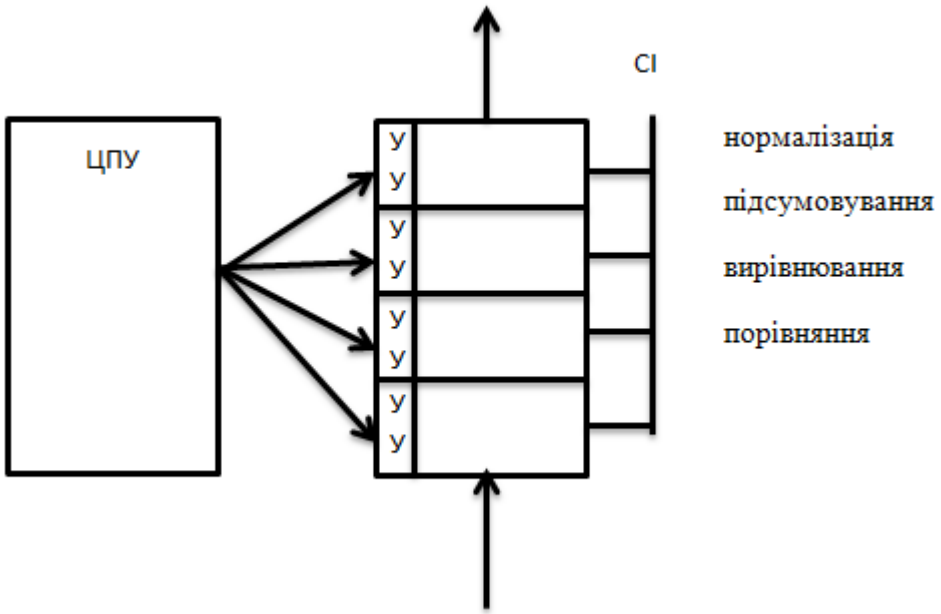


Рис. 49 Структурна схема MISD

STAR – 100 (100-100 млн. flops).

Конвеєр магістраль

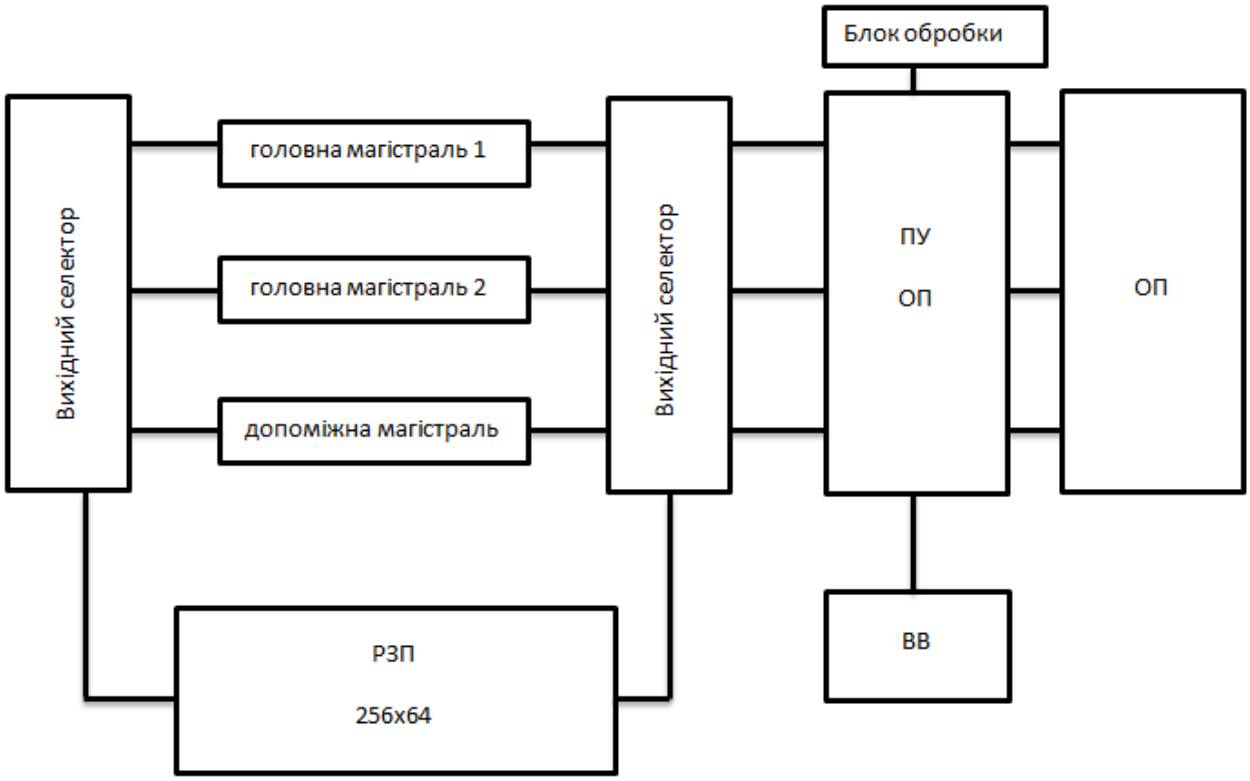
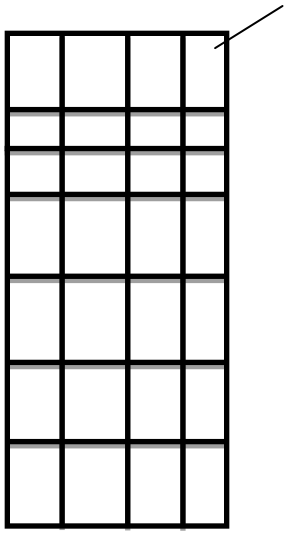
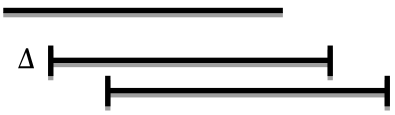


Рис. 50

Магістраль може ділиться на 2 частини (64 біт і 2 = 32бит)

Пам'ять (розділена на 32 модуля). Модулі теж працюють конвеєрним способом.





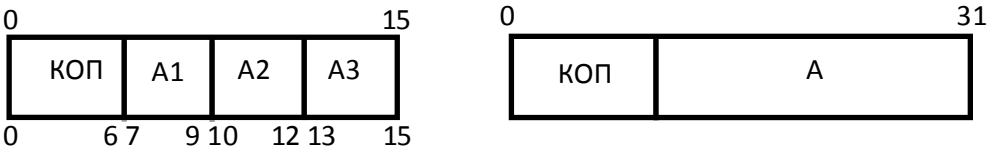
$$\Delta = 1.28 : 32 = 40\text{мс} : 16 = 2.5\text{ мс},$$

16 – кількість 32 – х розрядних слів (512: 32: 16)

Б – Буфери.

РОН – 8 64 регістр загально призначення.

Основний формат команд



τ=12.5 мс. P=80 млн flops

Тут використаний конвеєр конвеєрів P=170 млн flops $P_{пиков.}$=260 млн flops

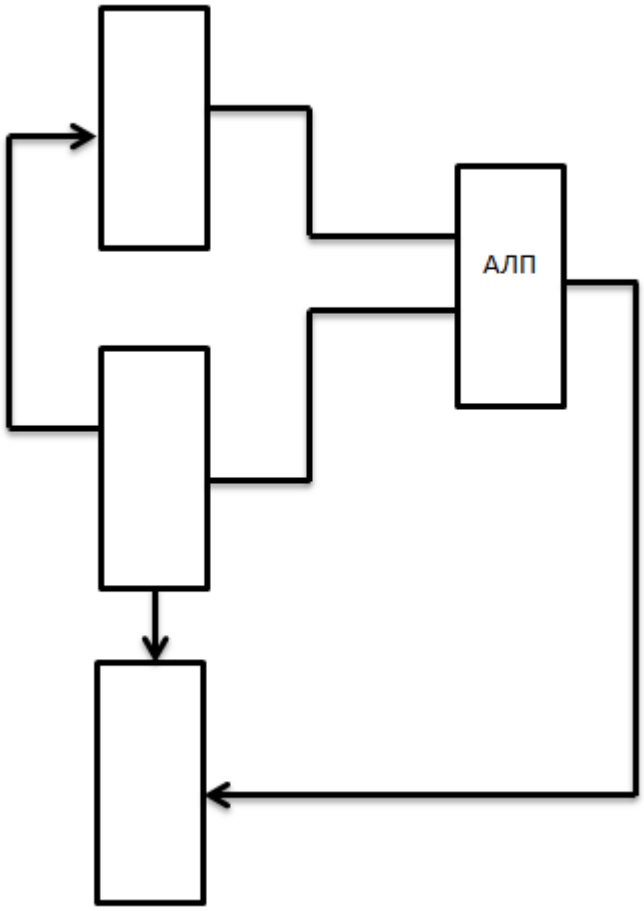


Рис. 51

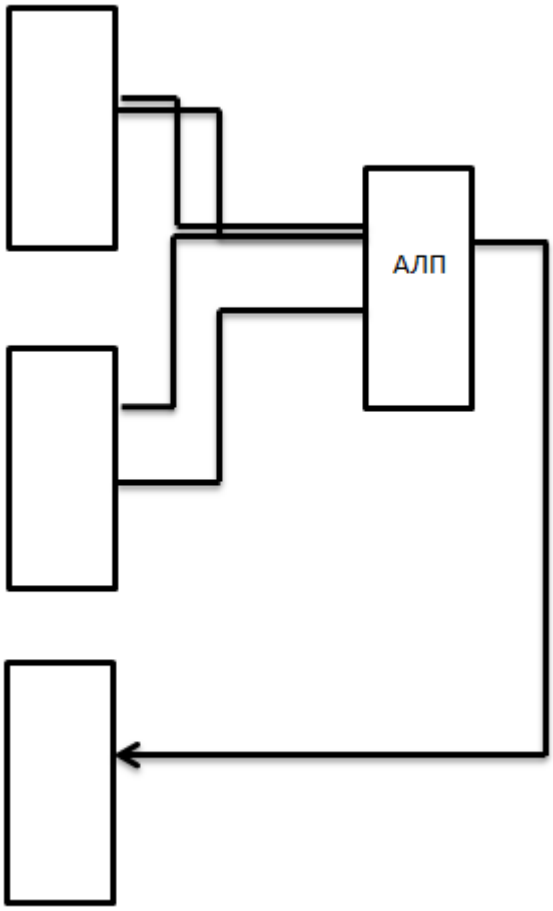


Рис. 52

Основні особливості Cray 1:

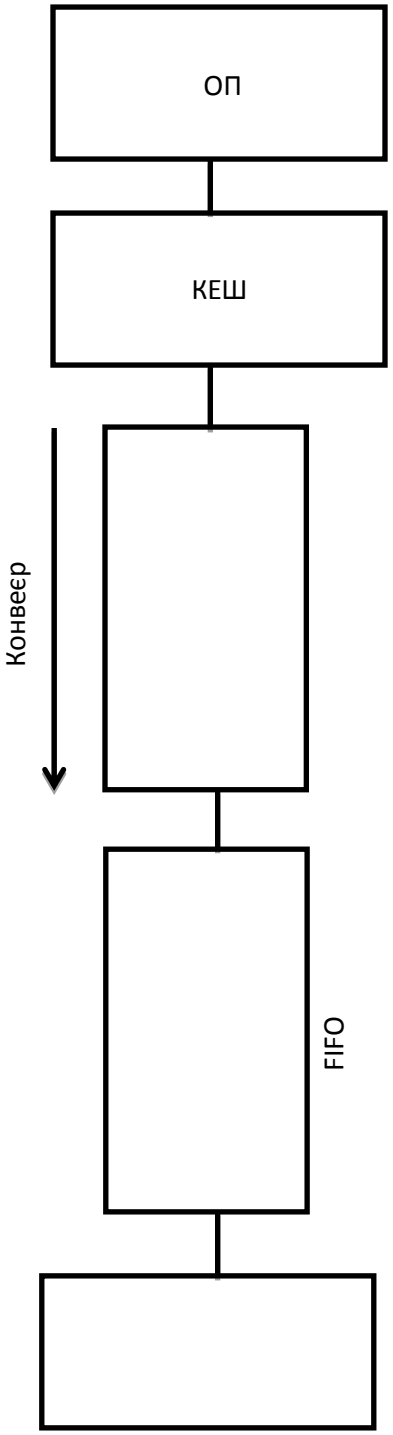
1) Ввід векторних операндів і реалізація векторних операцій.

2) Ввід між ОП і ЦП буферних регістрів володіють безліччю функцій кеш-пам'яті.

3) Реалізація конвеєра конвеєрів.

На ряду з тими конвеєрами якими розглянули (конвеєрами операцій), частина виділяють конвеєр команд і макроконвеєр.

Конвеєр команд



Формування лічильника адреси команд

Формування адреси команди яка буде виконуватися

Вибірка адреси команд, розпакування, формування

Використання адреси 1-го операнда

Вибірка 1-го операнда, - / / - 2-го, ...

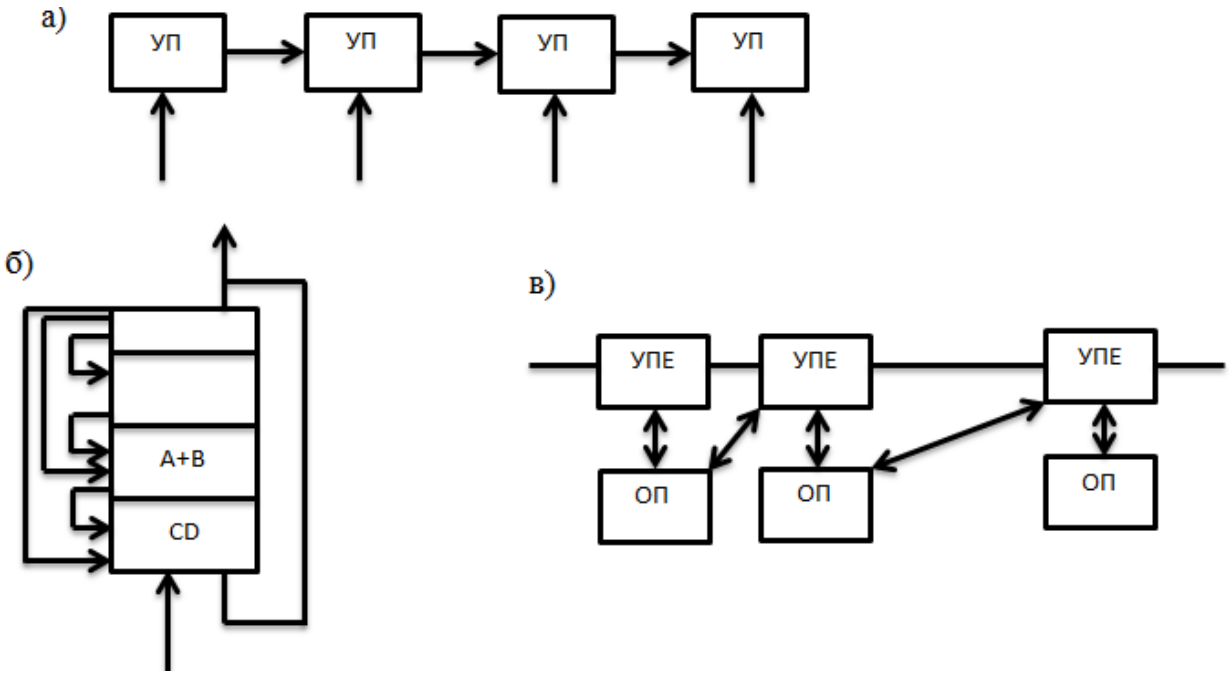


Рис. 53 ПЕ: а) макро; б) вертикальні; в) горизонтальні.

Такі ПЕ часто називаються макроконвеєрами.

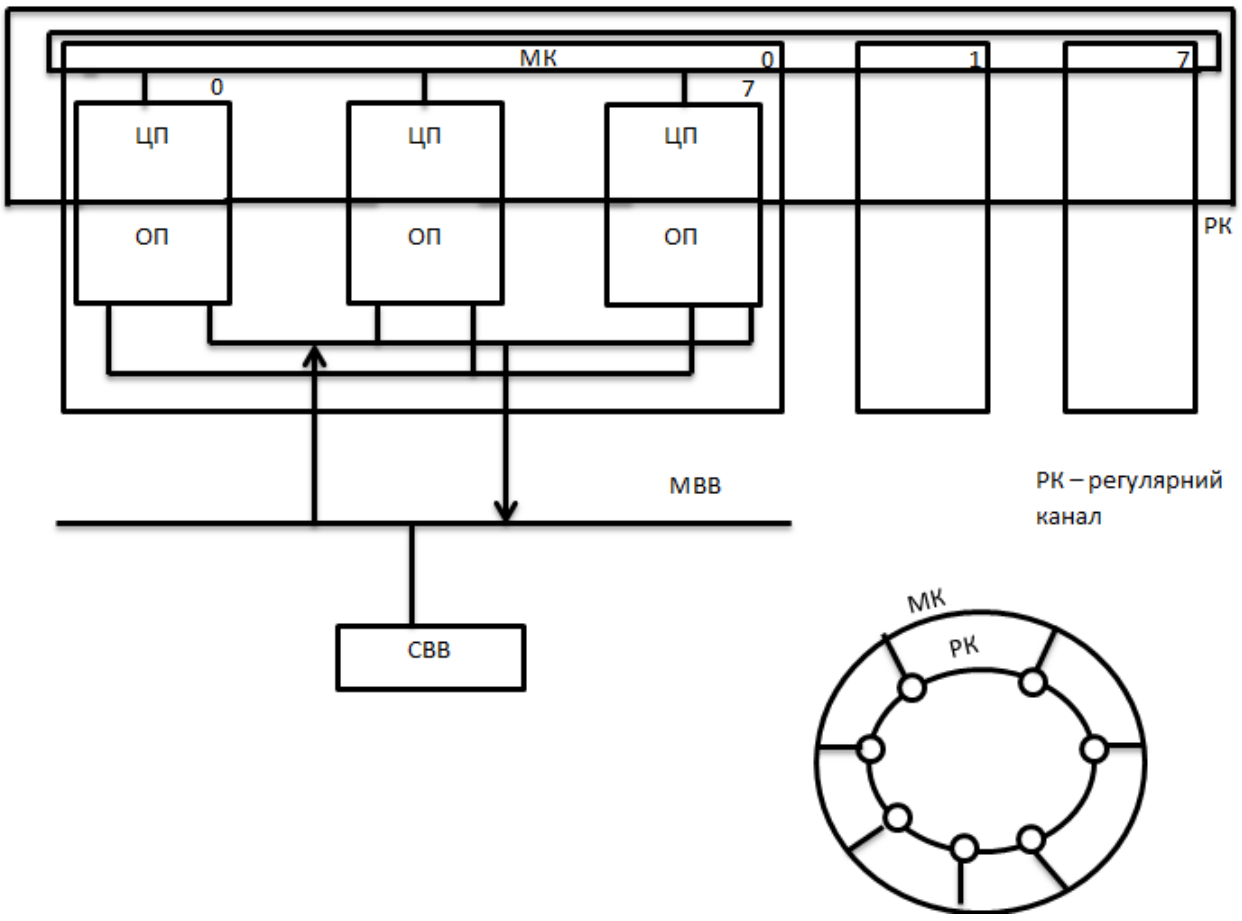


Рис. 54

$P_0 = 200$ млн $flops$

BSP

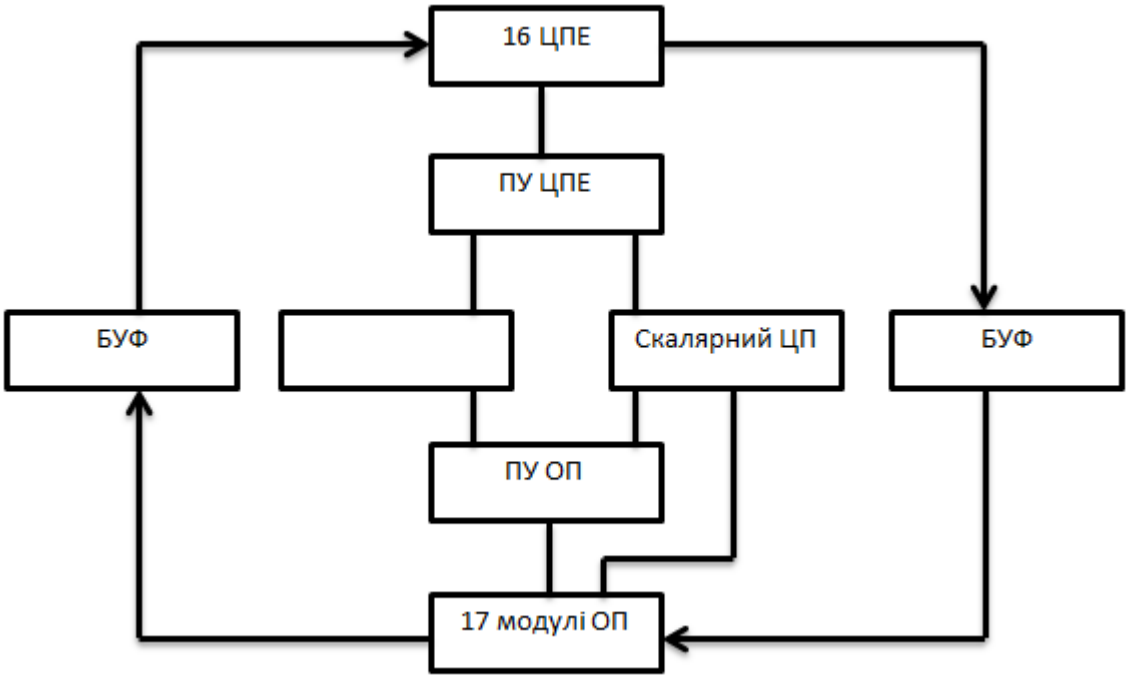


Рис. 55

25 млн flops

Два види розпаралелювання: матричне та конвеєрне.

Слідом за STAR-100 – ASC

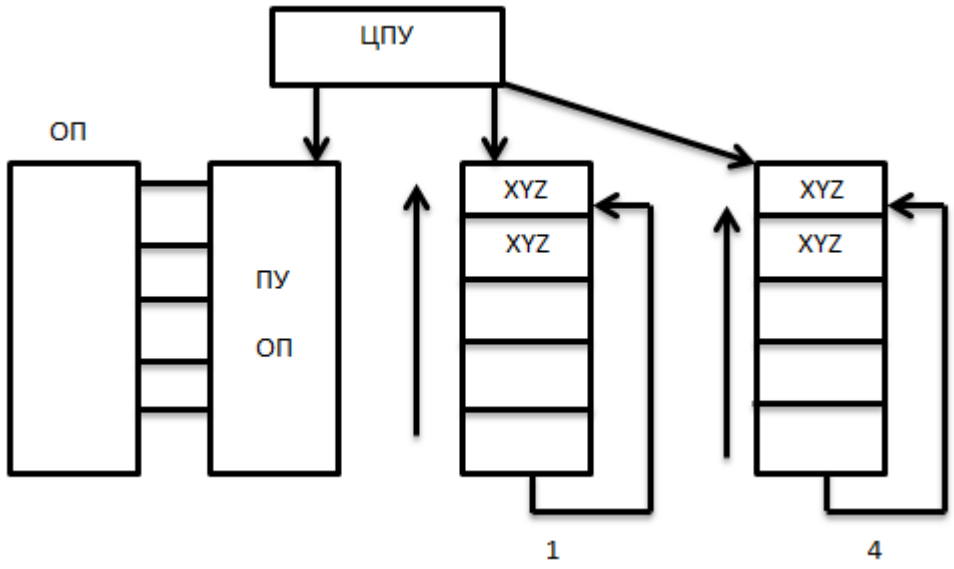


Рис. 56

$P_0 = 100$ млн $flops$, τ=40 мс.

MIMD

Будемо розрізняти наступні варіанти MIMD:

1) слабозв’язані
   Варіант, подібний до четвертого структурного покоління, в якому питання розпаралелювання вирішуються на рівні завдань, а питання взаємодії на рівні загальних файлів.
2) сильнозв’язані.

А) Мультипроцесорні системи, тобто системи в яких питання взаємодії вирішуються на основі ОП яка поділяється між усіма ПЕ. Загальним тут також є і зовнішній пристрій і канали вводу / виводу.

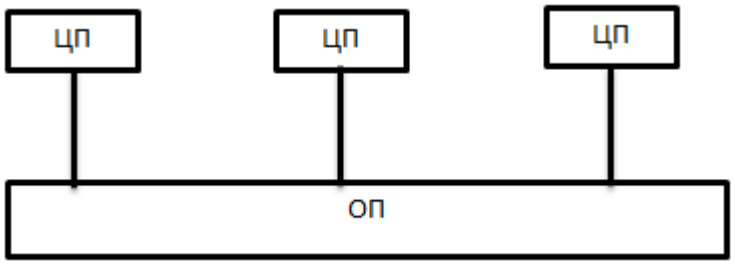


Рис. 57 Структурна схема мультипроцесорної системи

Основною перевагою такої системи є загальний адресний простір, що істотно впливає на розробку і реалізацію паралельних програм і взагалі задачу паралелізму. Тут реалізація паралелізму вирішується на рівні процесів.

Основним недоліком такої системи є швидке збільшення числа конфліктних ситуацій при зверненні до ОП, що створює великі черги і тривале очікування для різних ЦП. За цим число ЦП істотно обмежується, як правило, не перевищує 16.

Б) Мультикомп’ютерні системи в яких кожен ЦП має індивідуальну пам'ять (можливо і відповідні зовнішні пристрої і всі питання взаємодії вирішуються на основі комутації систем (мереж))

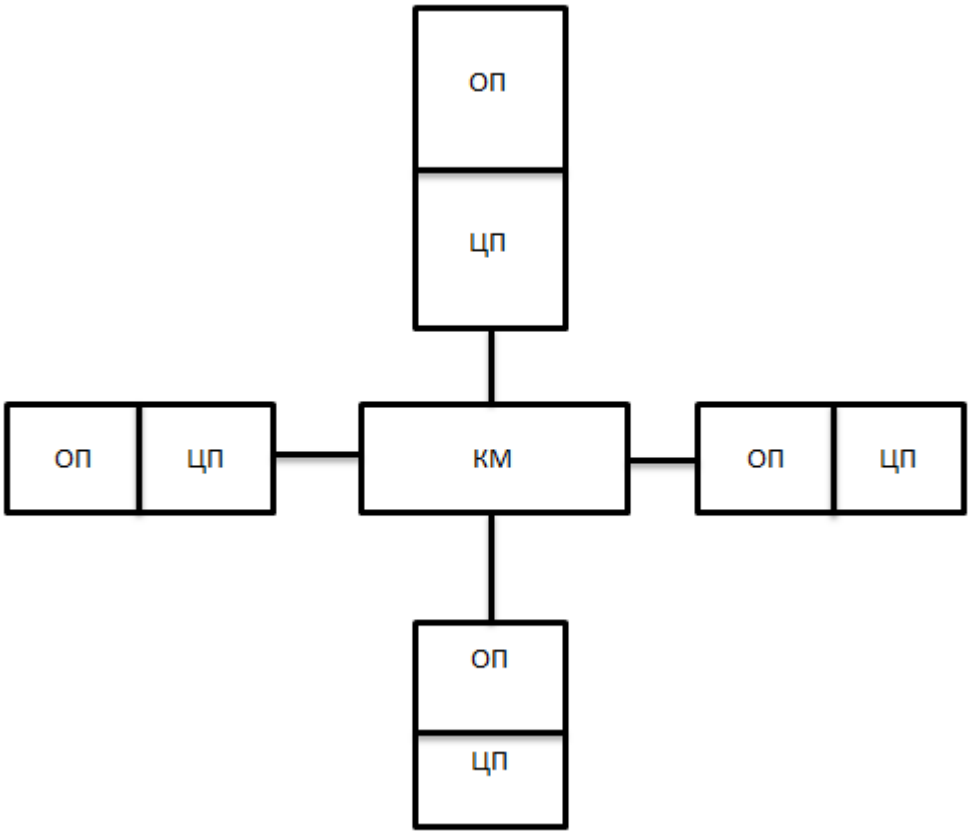


Рис. 58 Структурна схема мультикомп’ютерної системи

Основним недоліком такої системи є значне ускладнення питань організації та комутації і найголовніше кожен ЦП має індивідуальний адресний простір. Все це суттєво ускладнює питання розподілу, призначення взаємодій процесорних елементів між собою і відповідно істотно ускладнює процес підготовки паралельних програм до виконання. Але так як тут питання взаємодії вирішуються на основі КМ, які володіють широкими можливостями, то збільшення числа процесорів тут практично необмежено, що є основним достоїнством таких систем.

Система NUMA (Non-Uniform Memory Access з неоднорідним доступом до ОП).
Звернення до будь-якого модулю ОП, але за різний час.

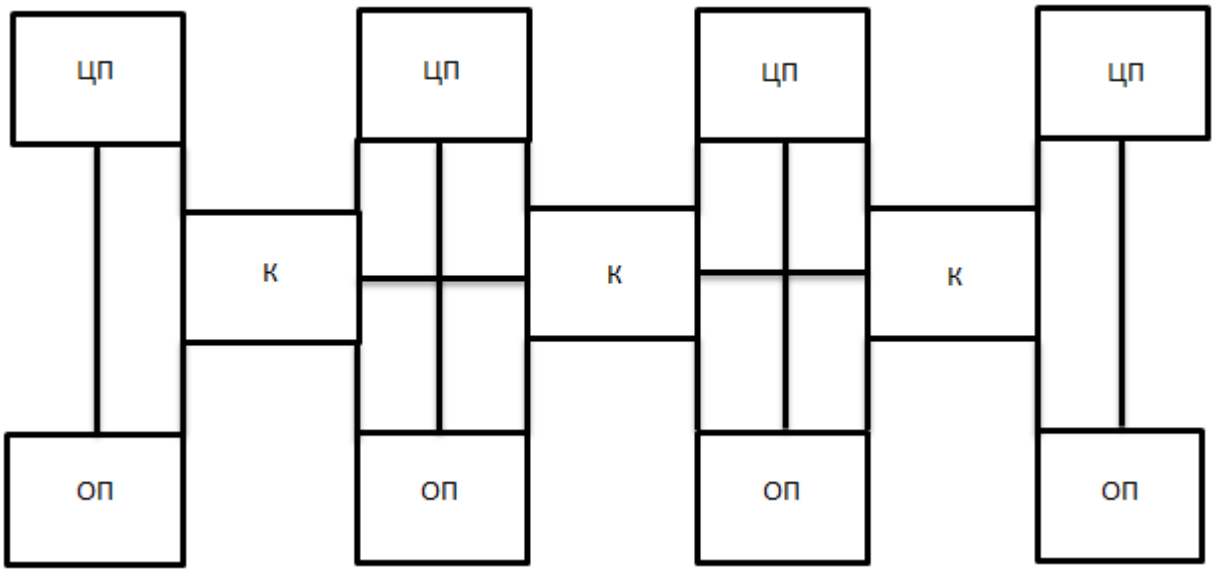


Рис. 59 Структурна схема системи NUMA

У даній ситуації імітується загальна ОП, але доступ до різних модулів виконується за різний час.

Кластер

Можна розглядати як деяку готову локальну мережу для якої розроблено спеціальне проміжне ПО яке дозволяє користувачеві реалізовувати безліч машин мережі, як окремо взяту машину, тобто користувач не бачить безліч машин які вирішують його завдання, а бачить одну дуже потужну машину. При розвитку кластерних систем зазвичай використовують канали зв'язку з великою пропускною здатністю (Наприклад Infiniband).

Grid системи

Це можливість кожному користувачеві отримати необхідну кількість ресурсів за рахунок використання безлічі машин глобальної мережі або internet у яких виставлено прапор «вільний».

Хмарні обчислення

Якщо в Grid кожен користувач може замовляти собі системи необхідної потужності, то при використанні хмарних систем користувач користується готовими «майстернями» в яких всі необхідні ресурси, включаючи спеців, вже є і які орієнтовані на визначення задачі. Користувач передає свої дані в «майстерню» і отримує результат.

Способи організації мультипроцесорних систем

1) Шинна організація.

2) Організація з матричним комутатором.

3) Багатопортова і багатошинна організація.

1)

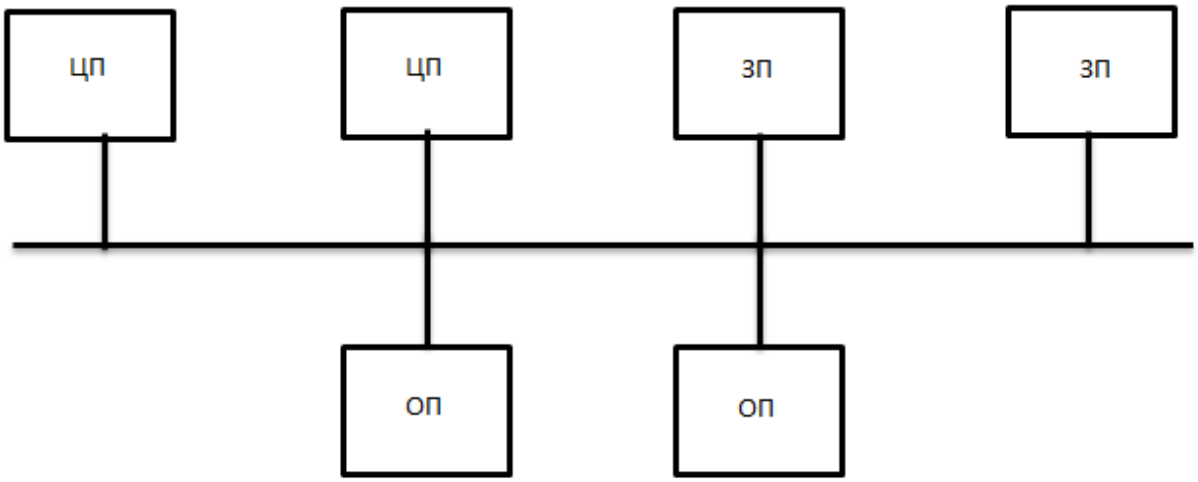


Рис. 60 Структурна схема системи із шинною організацією

Переваги:

1) Простота організації.

2) Простота нарощування елементів.

Недоліки:

1) Швидка насичуваність системи (системи з комутацією в часі, тільки 2 елементи можуть бути пов'язані між собою).

2) Низька надійність. Через дефекти шини система стає непрацездатною. Тому шина зазвичай робиться пасивною, а вся логіка взаємодії реалізується в елементах.

2)

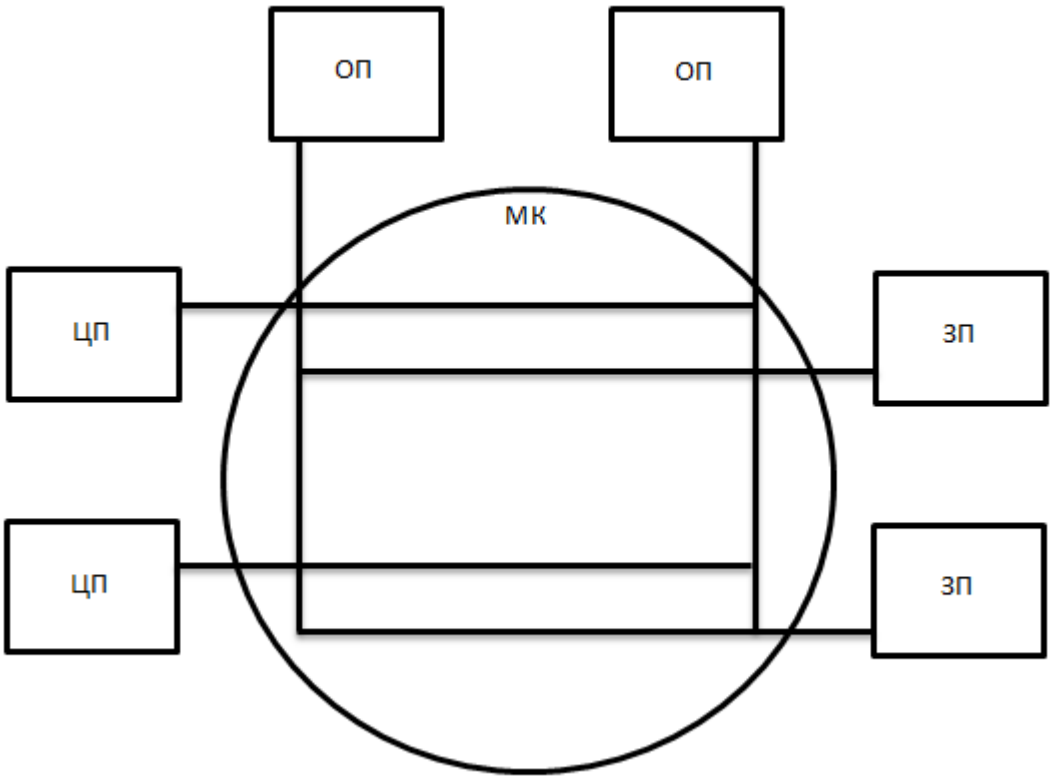


Рис. 61 Структурна схема система з матричним комутатором

Переваги:

1) Можливість реалізації безлічі каналів зв'язку одночасно (просторова комутація).

2) Простота нарощування елементів в межах розмірності комутатора.

Недоліки:

1) Складність, а отже і велика вартість комутаторів, які повинні вирішувати тут не тільки функції комутації, але й вирішення конфліктів.

3)

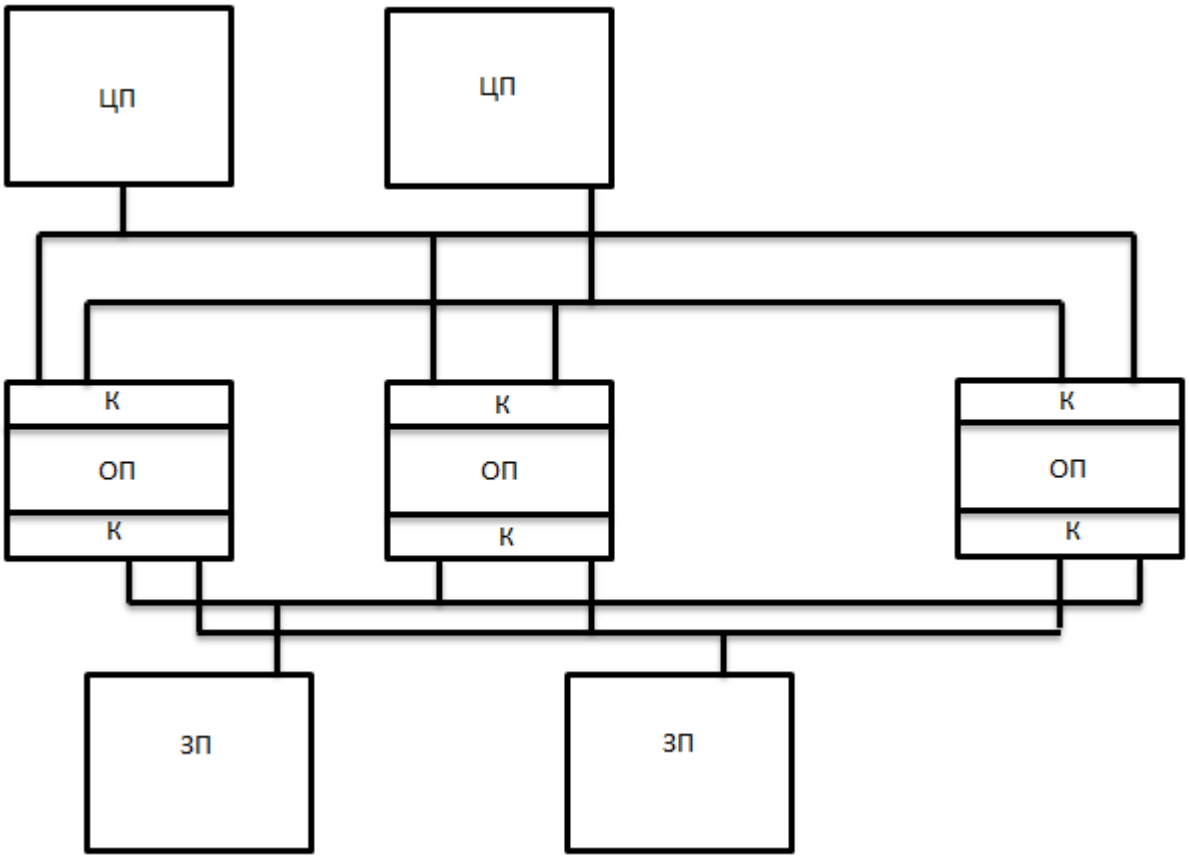


Рис. 62 Структурна схема системи з багато портовою і багато шинною організацією

Це певний проміжний варіант між першими двома варіантами, з одного боку може бути, 1 канал передачі одночасно, з іншого боку число таких каналів досить обмежено. Це пов'язано з тим що вартість багато-портової пам'яті істотно вища звичайної пам'яті, кількість портів обмежена 6-7.

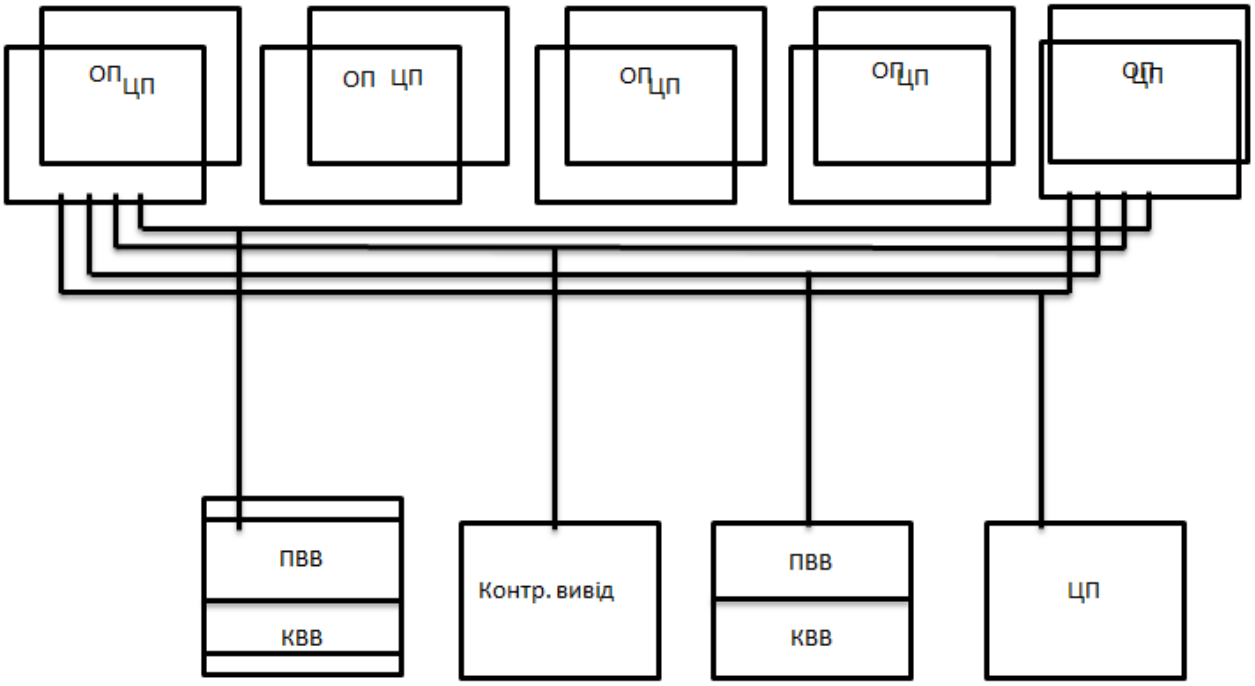


Рис. 63 Структурна схема системи **UNIVAC**

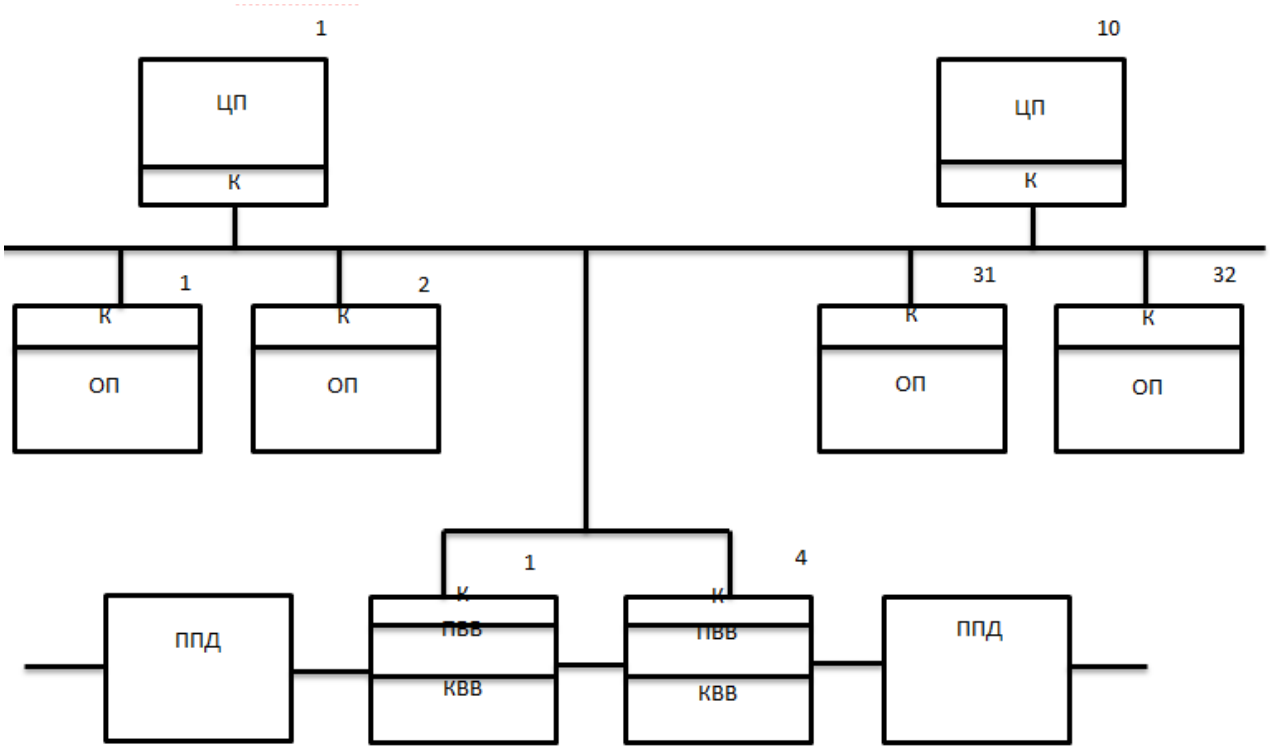


Рис. 64 Структурна схема системи **Ельбрус**

Організація ОП.

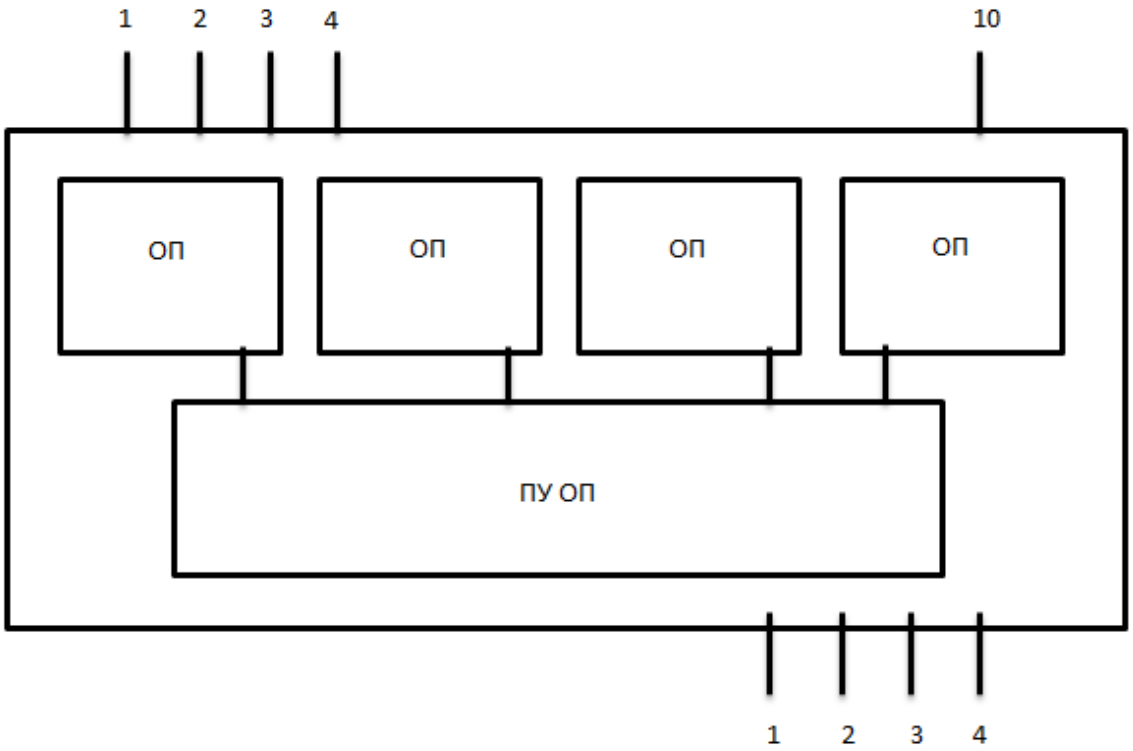


Рис. 65

Особливості:

Це машина високого рівня, тобто асемблер цієї машини була мовою високого рівня, який потім інтерпретувався в машинний код (виконувалася покрокова трансляція, в період етапу виконання програми).

При проходженні програм через машину найбільш важливим етапом слід вважати етап виконання (саме цей етап). Але в машині високого рівня питання компіляції (покрокової трансляції) переносяться саме на етап виконання, тим самим істотно сповільнюється реалізація виконання відповідної програми. Єдиним методом боротьби з цим явищем є широка апаратна підтримка при виконанні інтерпретації. У зв'язку з цим машина Ельбрус потребувала величезні кошти для вирішення інтерпретації апаратними засобами. Це і призвело до надзвичайно високої вартості цієї машини, а також і переосмислювання основних підходів і підвищенню продуктивності. Починаючи з Ельбрус і Burroughs з'явився новий напрямок RISC-архітектури, замість CISC.

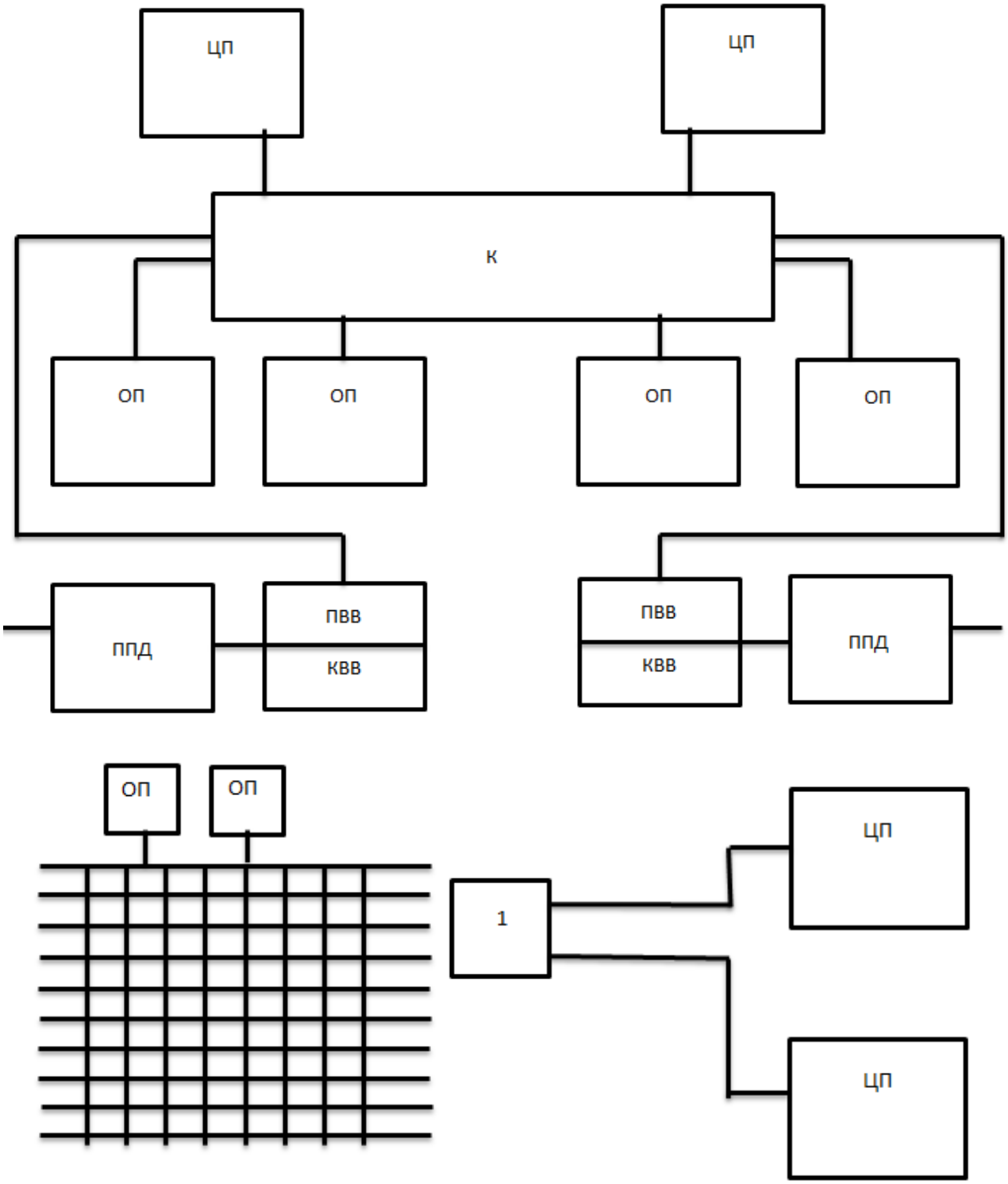
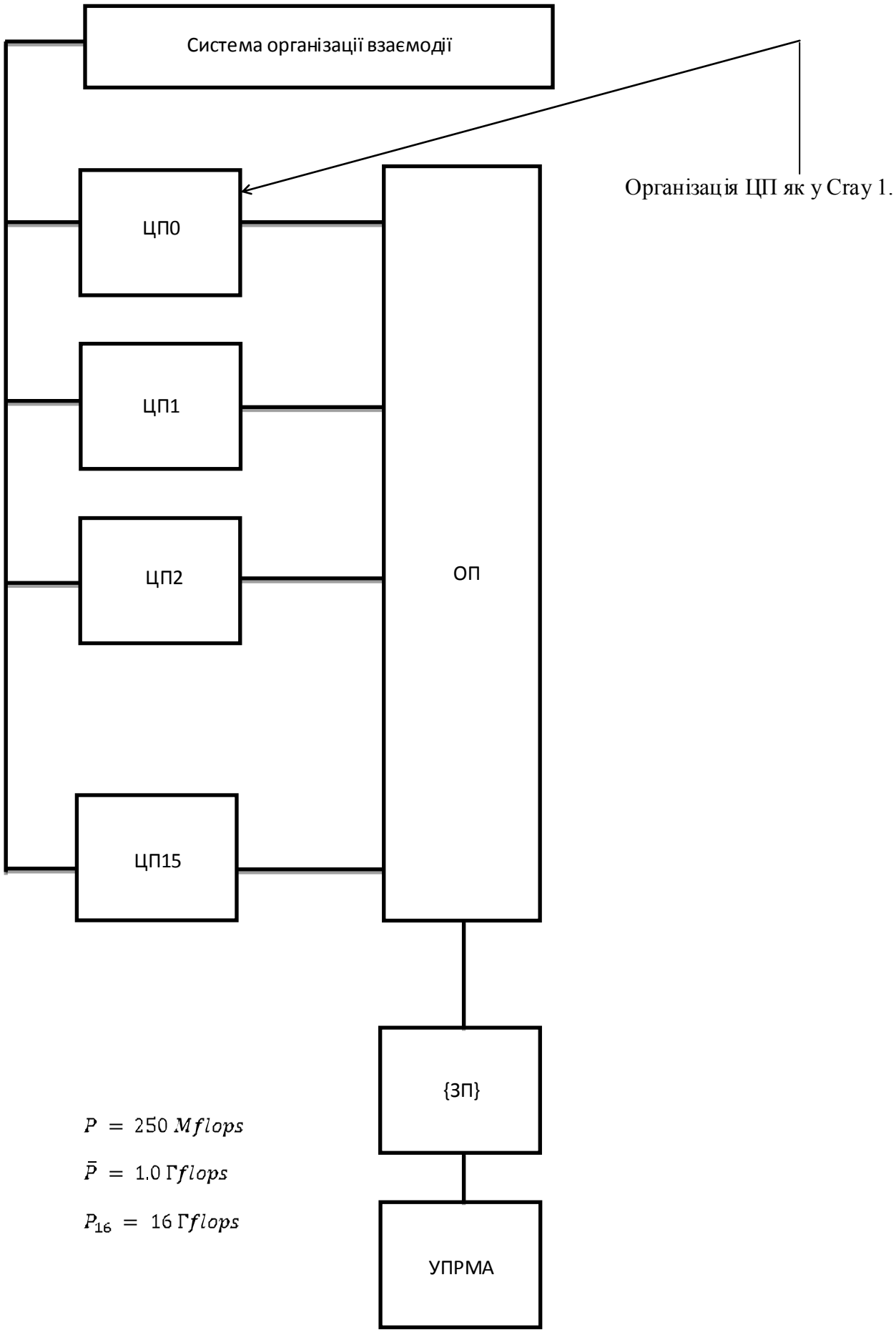


Рис. 66 Система **Burroughs**

CRAY 90



Організація ЦП як у Cray 1.

$P = 250\ Mflops$

$\bar{P} = 1.0\ Гflops$

$P_{16} = 16\ Гflops$

Топологічна організація мультикомп’ютерних систем

Під топологією будемо розуміти безліч елементів і зв'язків між ними без урахування будь-яких інших структурних засобів крім засобів існування і зв'язності.

G(x, y). x- множина вершин та вузлів.

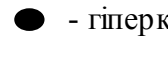
У - множина ребер.

Ми будемо розглядати графи у яких відсутні крайні зв'язки та внутрішні цикли, і ступінь топології яких перевищує кратність 2.

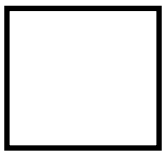
Метрика топологічних побудов

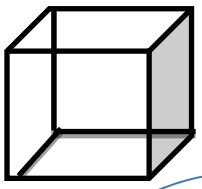
1) Масштабованість, тобто можливість необмеженого збільшення вузлів при збереженні топологічних властивостей системи.

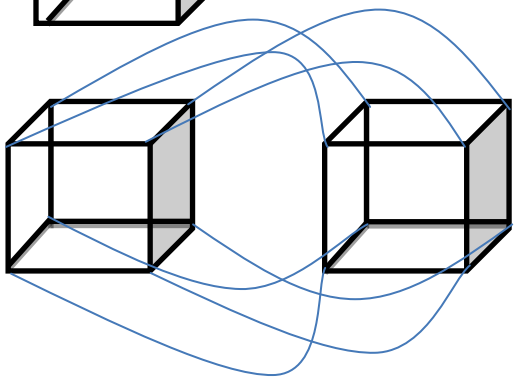
Гіперкуби різних порядків

 - гіперкуб нульового порядку.

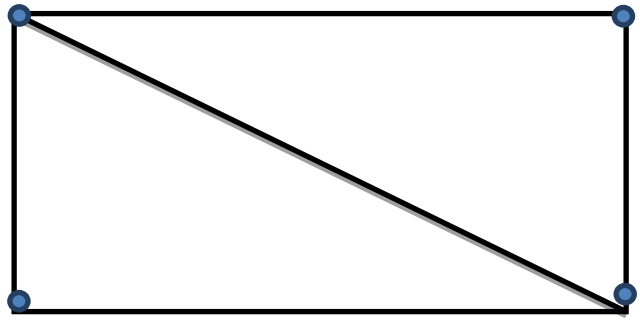
 - гіперкуб першого порядку.

 - гіперкуб другого порядку.

 - гіперкуб третього порядку.

 - гіперкуб четвертого порядку.

Ступінь топології - це максимальне число ребер інцидентних одній вершині.



S = 3

В основному визначає вартість і тому даний параметр підлягає оптимізації

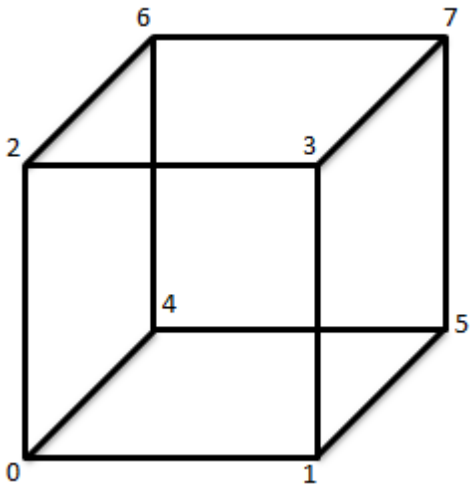
Оптимальний S = 4 ÷ 6

Локальна зв'язок - це мінімальне число ребер, інцидентних одній вершині.

Діаметр - це мінімальна відстань між найбільш віддаленими вершинами.

Середній діаметр, який в основному визначається пропускними здатностями системи - це деяка усереднена відстань між вершинами графа.

$$\overline{D} = \frac{(\sum_{i=0}^{N-1}\sum_{j=i+1}^{(N+i-1)\,mod\,N} d_{i,j})}{N(N-1)}$$ — для симетричних графів(фігур)

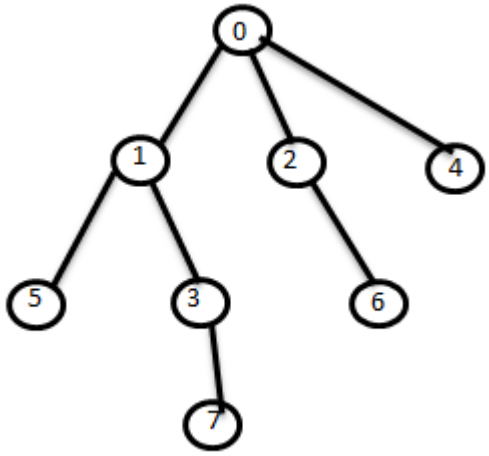


Топологічний трафік - це не реальна, а номінальна ступінь завантаження ребер графа.

3 * 1 + 3 * 2 + 1 * 3 = 12

$$\overline{D} = \frac{12}{7}$$

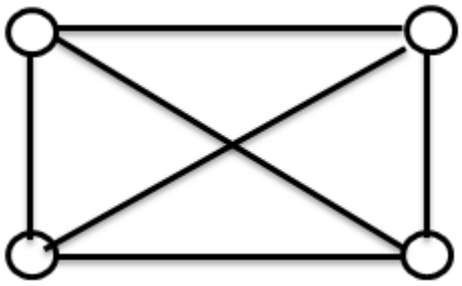
$T = \frac{N * \overline{D}}{R} = \frac{2 * N * \overline{D}}{N * S} = \frac{2 * \overline{D}}{S}$, R — реальна кількість ребер.



$T \to 1$

$T_{куб} = \frac{2 * 12}{7 * 3} = \frac{8}{7}$

$T = \frac{2 * 1}{1024} \to \frac{1}{512}$, повнозв'язна система з 1024 вершинами



$SD \to min$

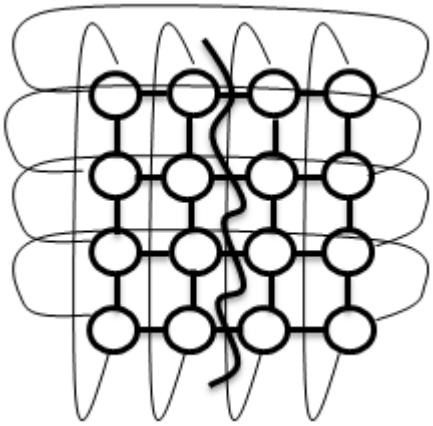
Якщо цей критерій однаковий для безлічі топологій, то вибираємо ту, де
$T \to 1$

Пересіченість топології.

Будемо розрізняти топології з фіксованою системою зв'язку і топологію з системою зв'язку, що реконфігуруються.

Введення комутаторів збільшує складність топології і її вартість і вносить додаткові затримки, тому найбільш часто використовують локальні комутуючі елементи, тобто між деякими вершинами, але досить часто використовується топологія з фіксованою системою зв'язку. Крім того питання масштабованості істотно ускладнюються в мережах зв'язку, що реконфігуруються ,так як кількість елементів обмежується параметрами комутаторів.

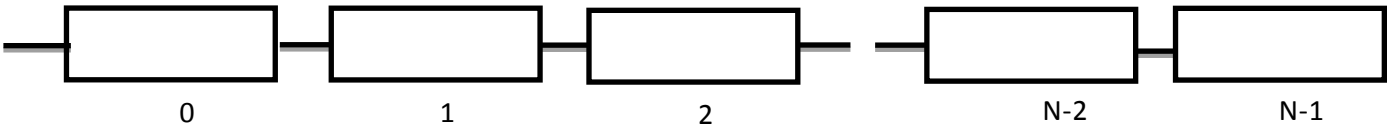
Глобальна зв'язність визначає кількість ребер які необхідно розрізати щоб розділити відповідну топологічну організацію на дві рівні множини.



Вартість топологічної організації визначається ступенем топології. Тому ступінь підлягає мінімізації.

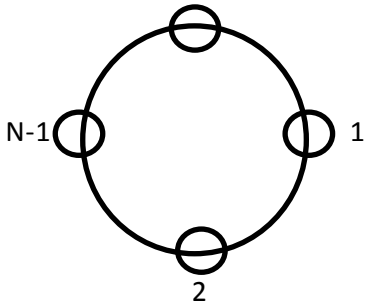
Типові топологічні організації

1)Лінійний ланцюг.



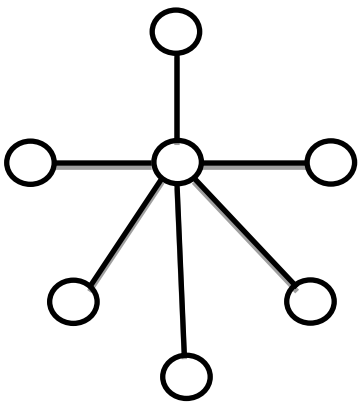
$$S = 2, D = N - 1$$

2)Кільце.



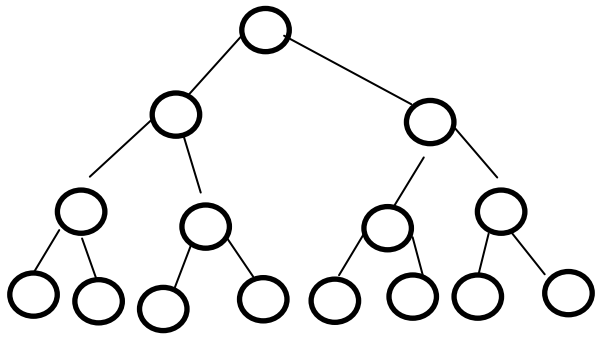
$$S = 2, D = \frac{N}{2}$$

3)Зірка.



$$S = N - 1, D = 2$$

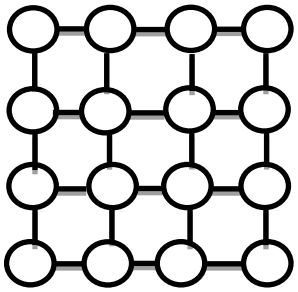
4)Дерево.



$S = 3$

$D = 2\log\frac{N+1}{2}$

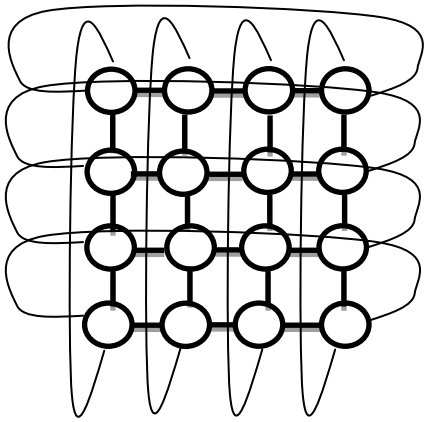
5)Решітка



$S = 4$

$D = 2(\sqrt{N} - 1)$

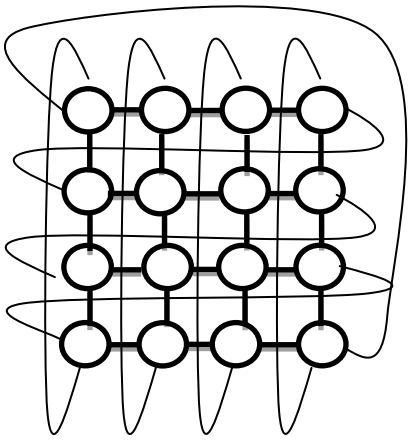
6)Меш



$S = 4$

$D = \left[2(\frac{\sqrt{N}}{2})\right]$

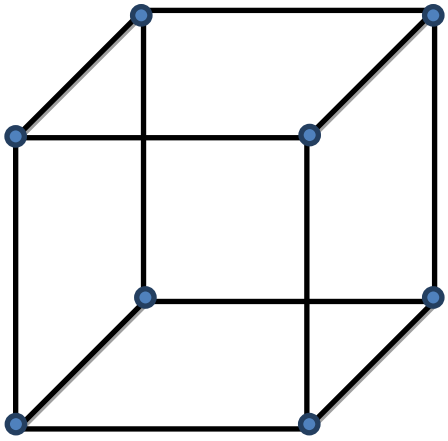
7)Хордова



$S = 4$

$D = \sqrt{N} - 1$

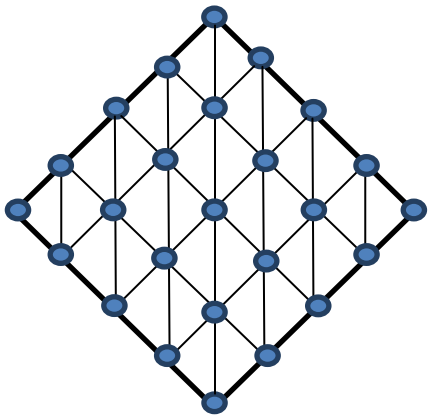
8)Гіперкуб



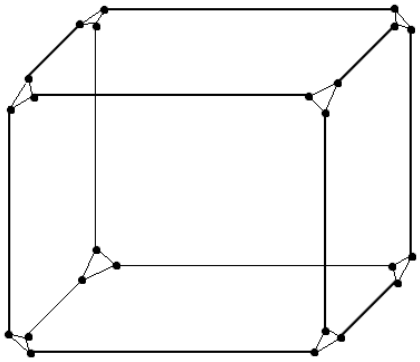
$S = \log_2 N$

$D = \log_2 N$

9)Систолічний



Способи мінімізації ступеня топологічних організацій.



N=24, S=3, D=6

Ієрархічний гіперкуб (гіперкуб у гіперкубі)

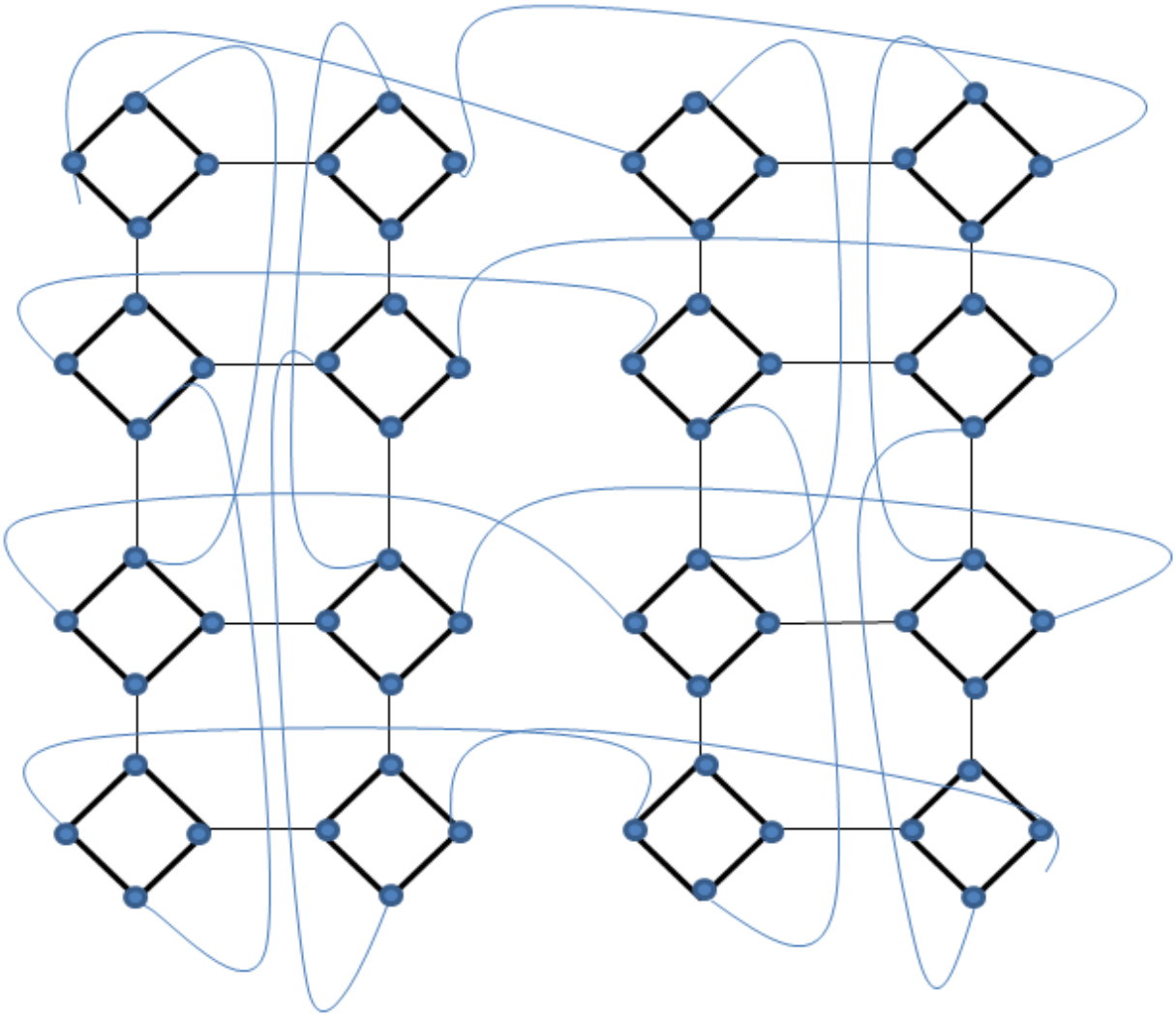


Рис. 67

Вбудовуємо в гіперкуб, гіперкуб порядку2^p

P=4

N=$2^{2^4+4} - 2^{20}$

S=5

Способи мінімізації діаметра топологічних організацій

Одним зі способів є такий який заснований на побудові топологічних організацій з високими або змішаними основами.

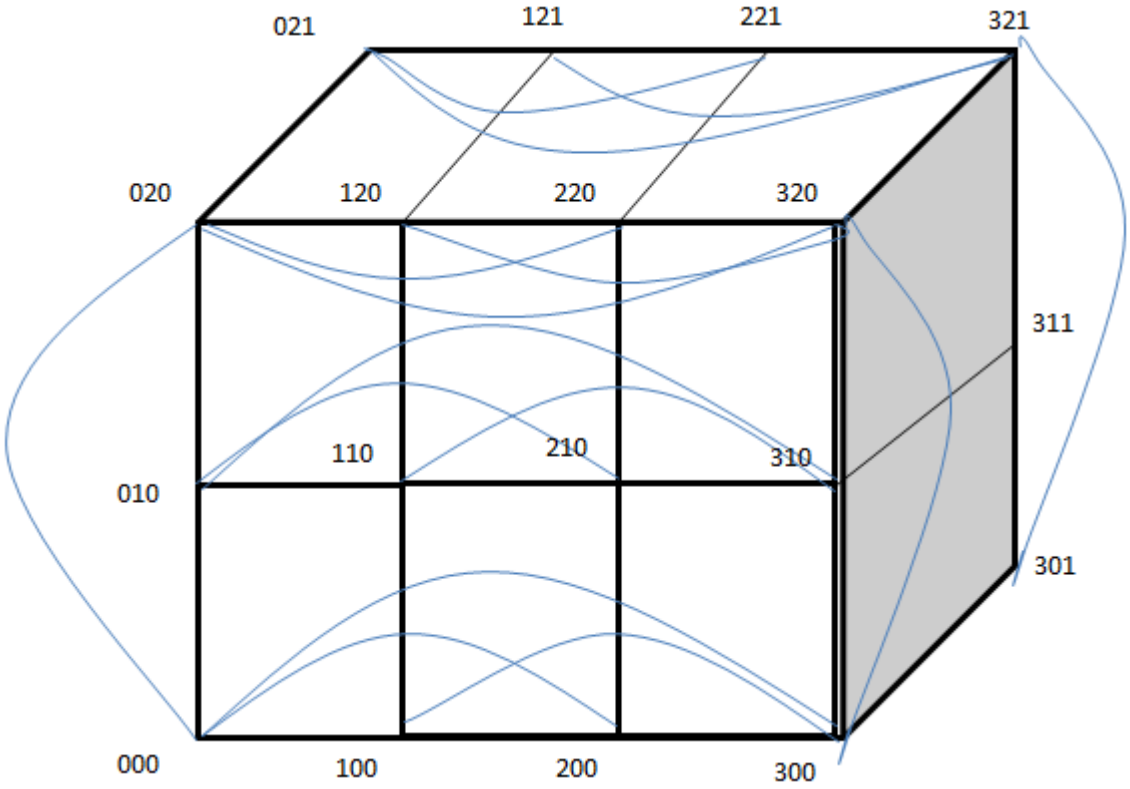


Рис. 68

Діаметр дорівнює кількості вимірів= 3

N=24

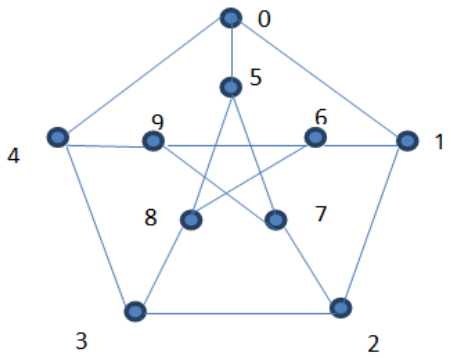
S=$\sum_{i=1}^{r} m_i - r$. m_i — основа системи числення

D*S=3*6=18.

2+3+4-4=6.

Другий підхід орієнтований на пошук деяких простих і найбільш ефективних топологічних організацій з наступним поверненням в квадрат і куб на основі декартового добутку.

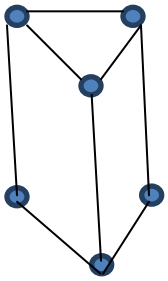
S =3, D=2, D*S=6



Для N=100:

S =6, D=4

Декартовий добуток



N=$N_1 * N_2$

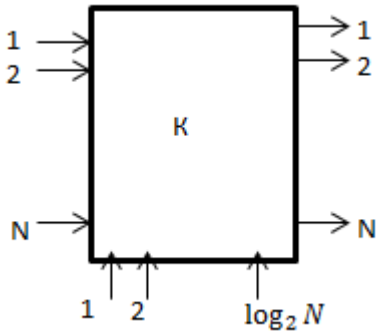
R=2*3+3*1=9

S=$S_1 + S_2$

D=$D_1 + D_2$

Комутовані системи

(мережі з системою зв'язку, яка реконфігурується).



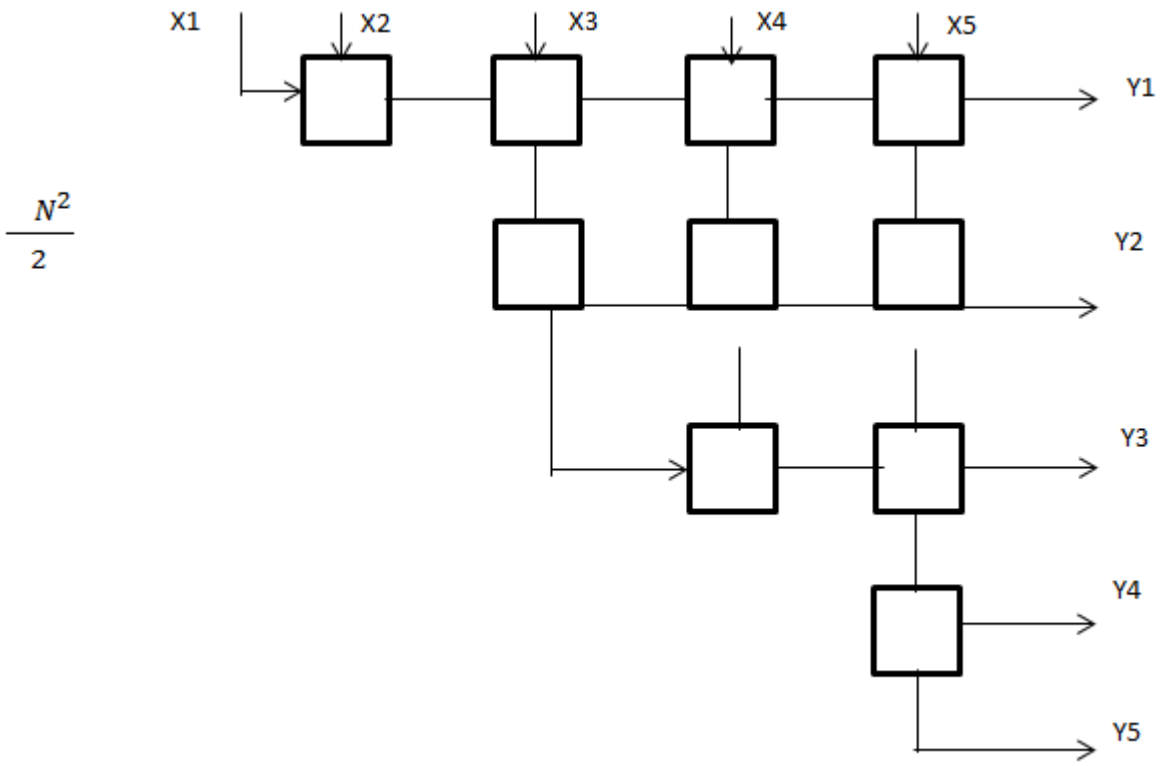


Рис. 69

Переваги:

Можливість створення неблокованих комутаційних мереж.

Недоліки:

1) Дуже велика кількість елементів для реалізації комутатора

Звідси→

2)Значний час на реалізацію $\frac{N^2}{2}$ комутації.

3) Кожен елемент має значну складність і вимагає як мінімум 3 тригера для утримування налаштуваннь.

4) Необхідність індивідуального налаштування кожного елемента.

Першою і основною реалізацією комутаторів є **мережа Клозе**.

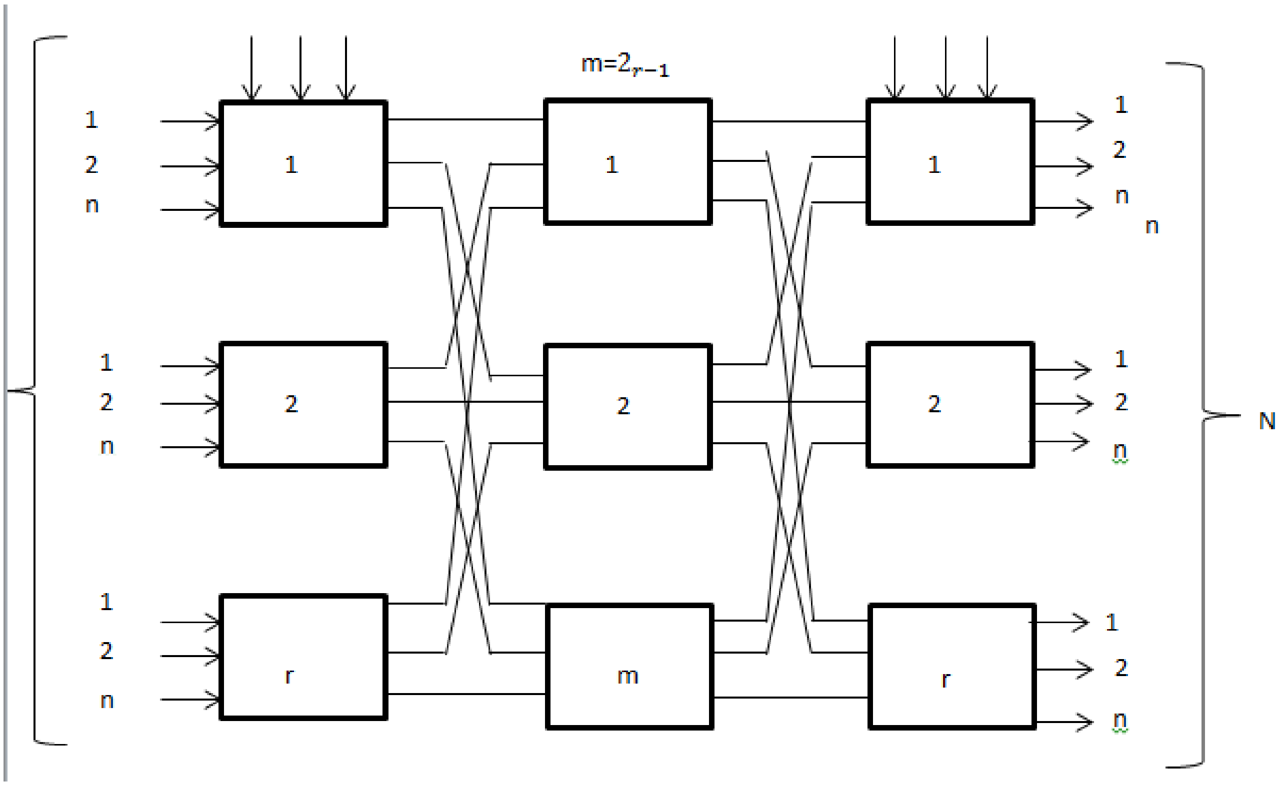


Рис. 70 Мережа Клозе

Заслуга Клозе полягає в тому що він показав можливість реалізації 3х крокової конструкції комутаторів і вивів відношення при реалізації якого з'явилася можливість створювати неблоковану мережу.

переваги:

1) Реалізація комутації за 3 кроки-малий час комутації.

2) Можливість налаштування для виключення блокувань.

недоліки:

1) Значна складність кожного комутатора.

2) Необхідність індивідуального налаштування кожного окремо взятого елемента.

Мережа Бенеша

Налаштування йде по одному біту

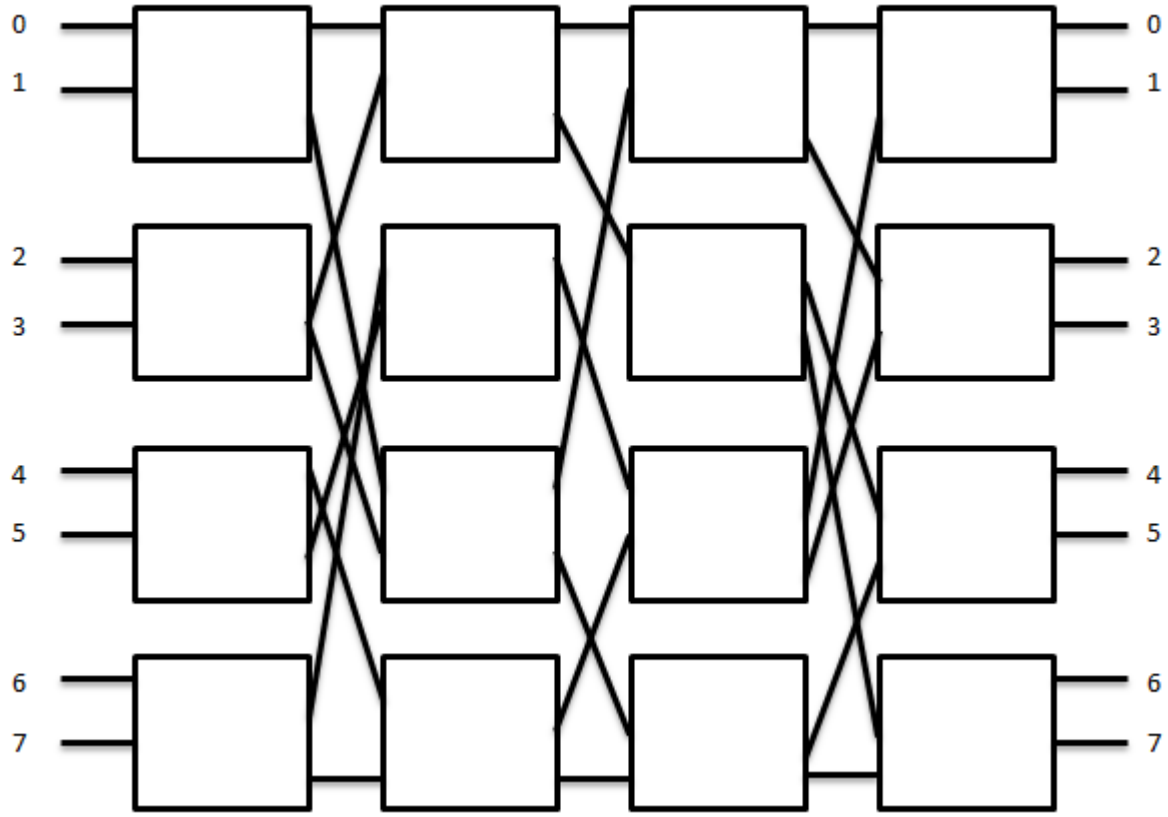


Рис. 71 Мережа Бенеша

переваги:

1) Можливість створення неблокованої мережі.

2) Використання досить простих елементів.

переваги:

1) Дуже велика кількість сигналів, що необхідні для реалізації.

2) Значні витрати (часу) на реалізацію комутації.

3) Необхідність в індивідуальній настройці.

4) Неоднорідність комутації між каскадами, що практично виключає реалізацію комутації в часі.

З появою різних проблемно-орієнтованих систем з'явилася доцільність використання в загальному випадку блокувань, комутаційних елементів які однак для багатьох процесів, завдань можуть виявитися неблокованими.

Схеми, що блокуються

Баньян

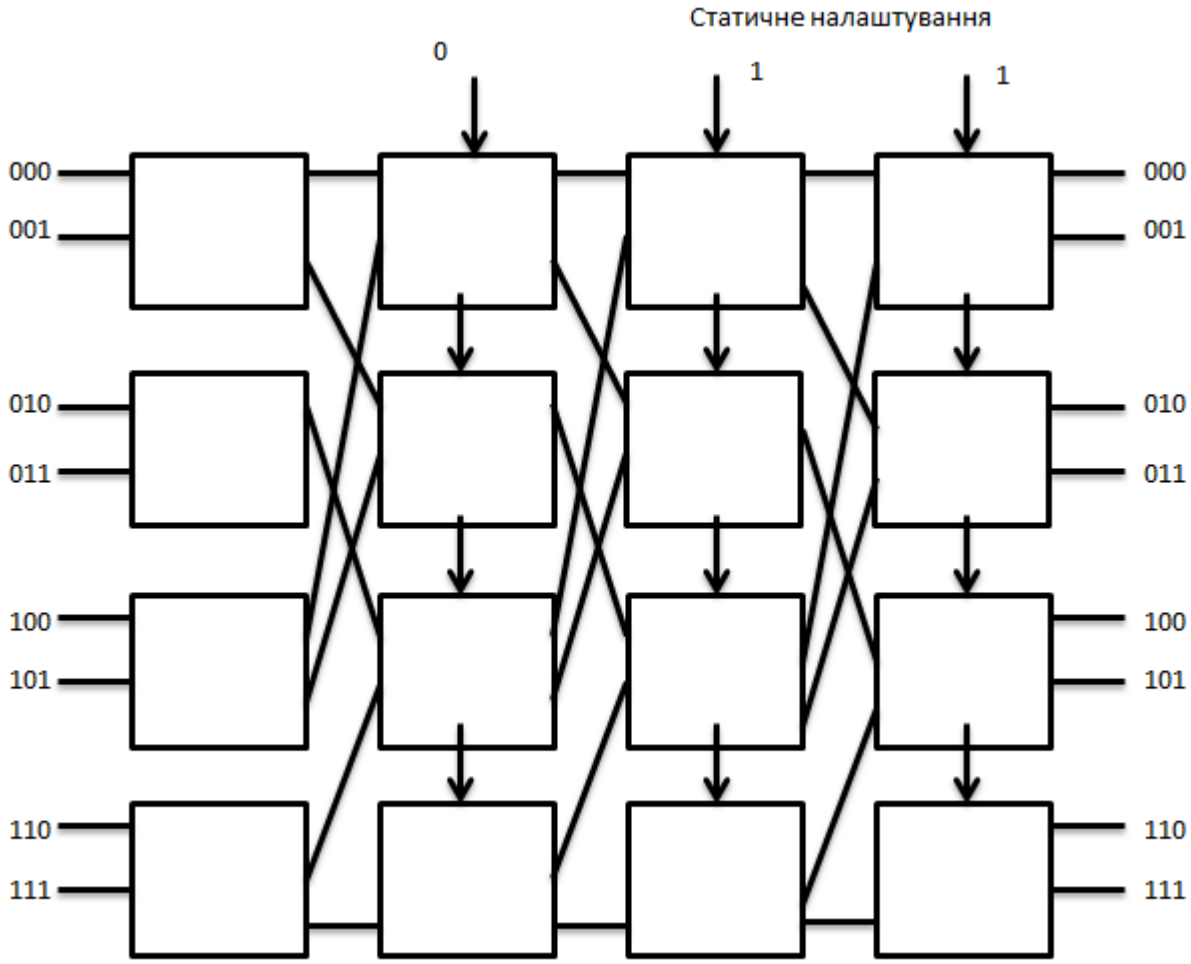


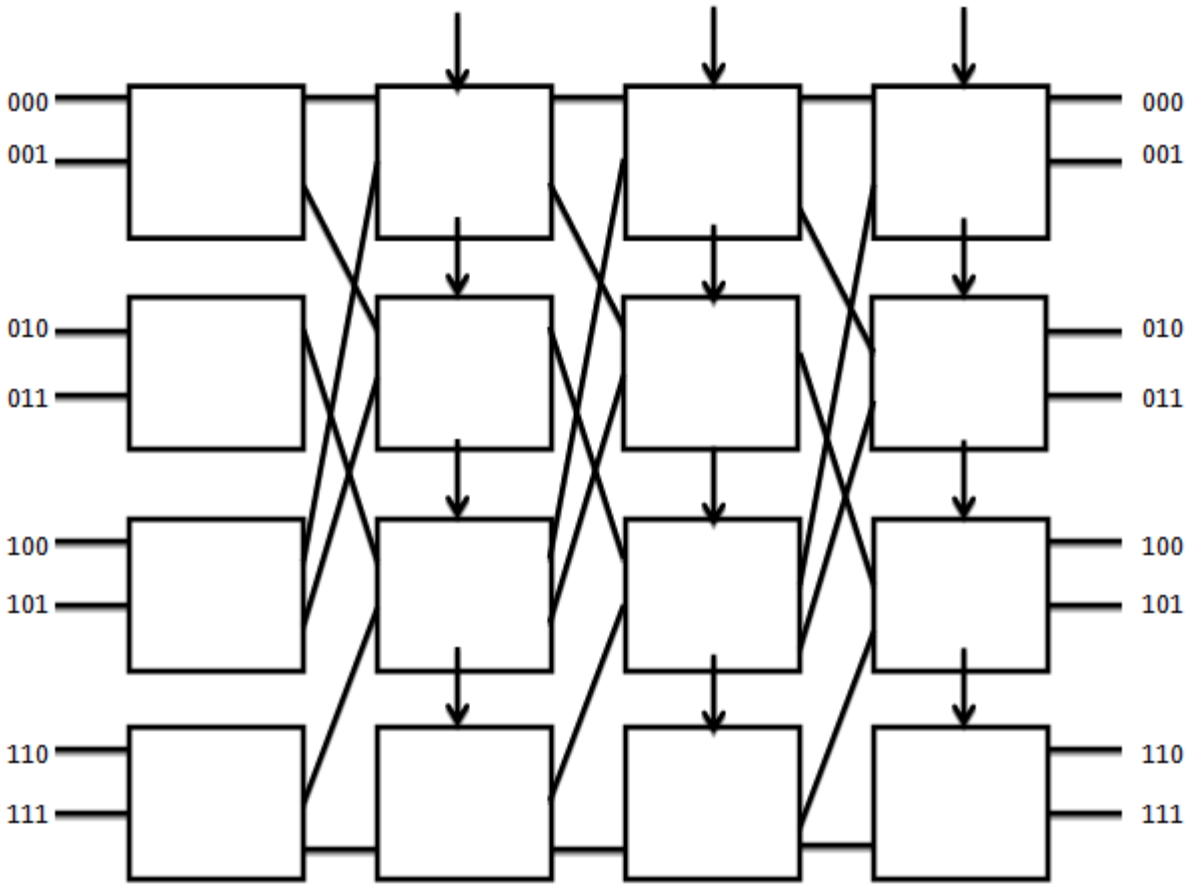
Рис. 72 Мережа Баньян

Недоліки:

1. Можливість блокування передачі.
2. Різнорідність зв'язків між каскадами, що істотно ускладнює можливість реалізації комутації в часі.

Переваги:

1. Удвічі менша кількість каскадів, а відповідно і елементів для реалізації комутації.
2. Можливість реалізації маршрутизації статично і динамічно, причому регулюється дуже просто.


000
001
010
011
100
101
110
111
000
001
010
011
100
101
110
111

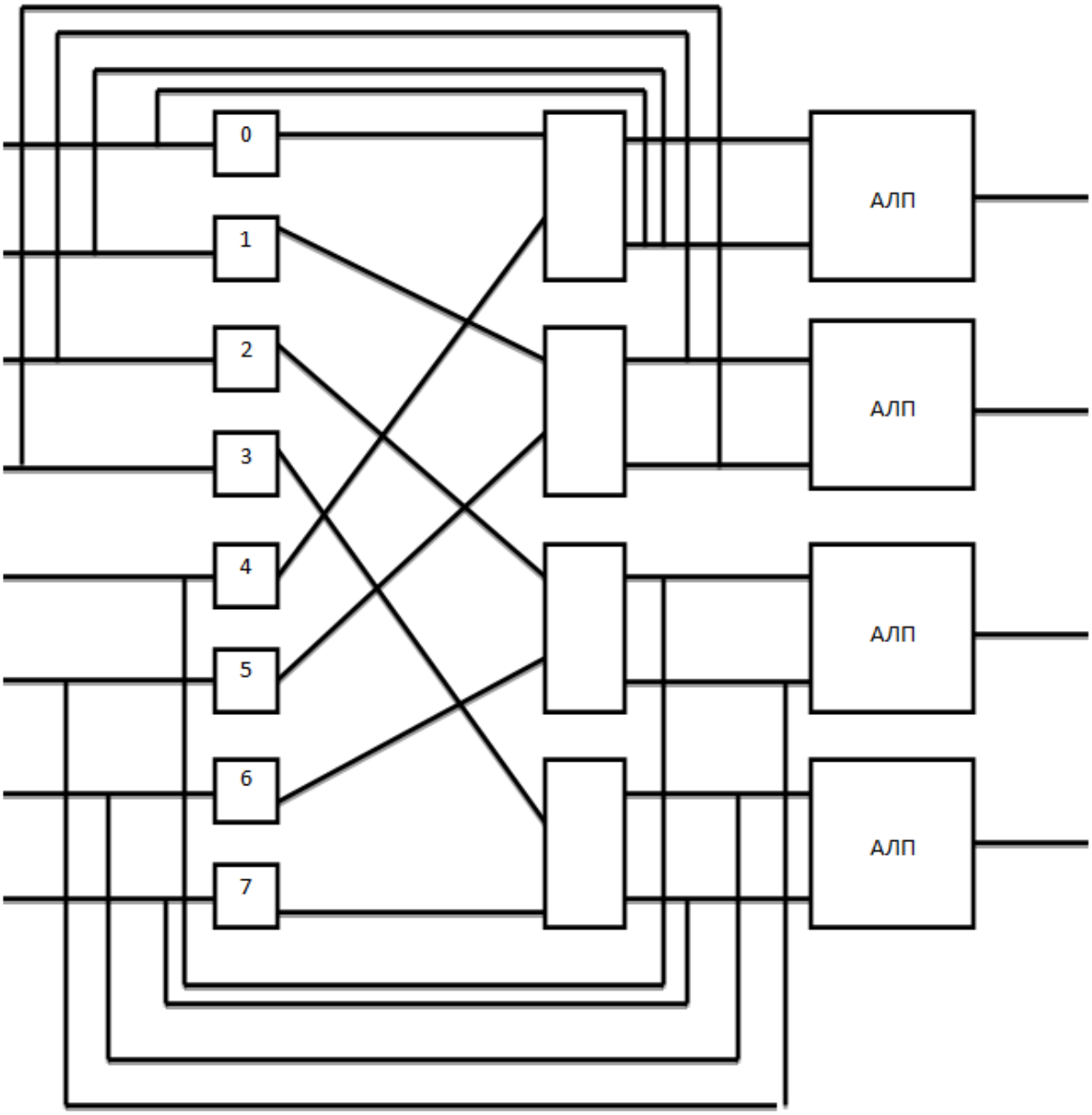


Рис. 73

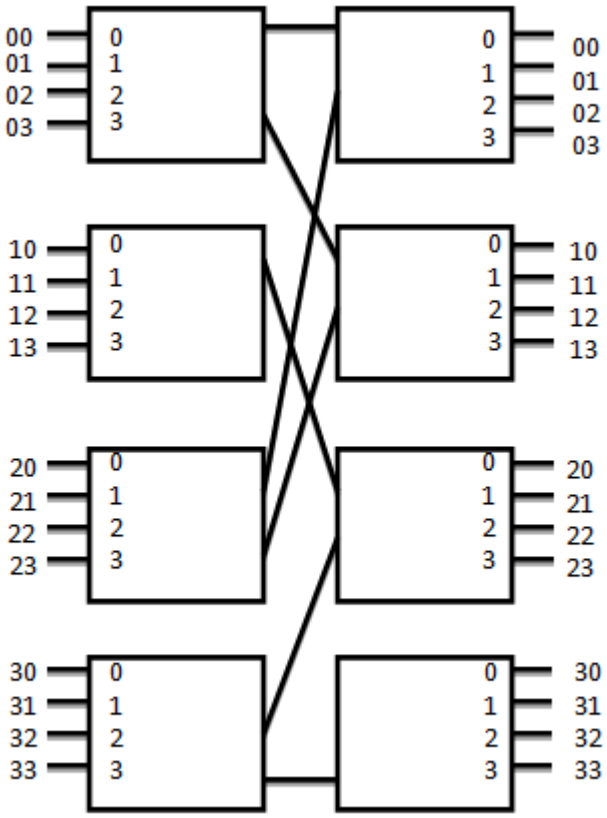


Рис. 74 Дельта

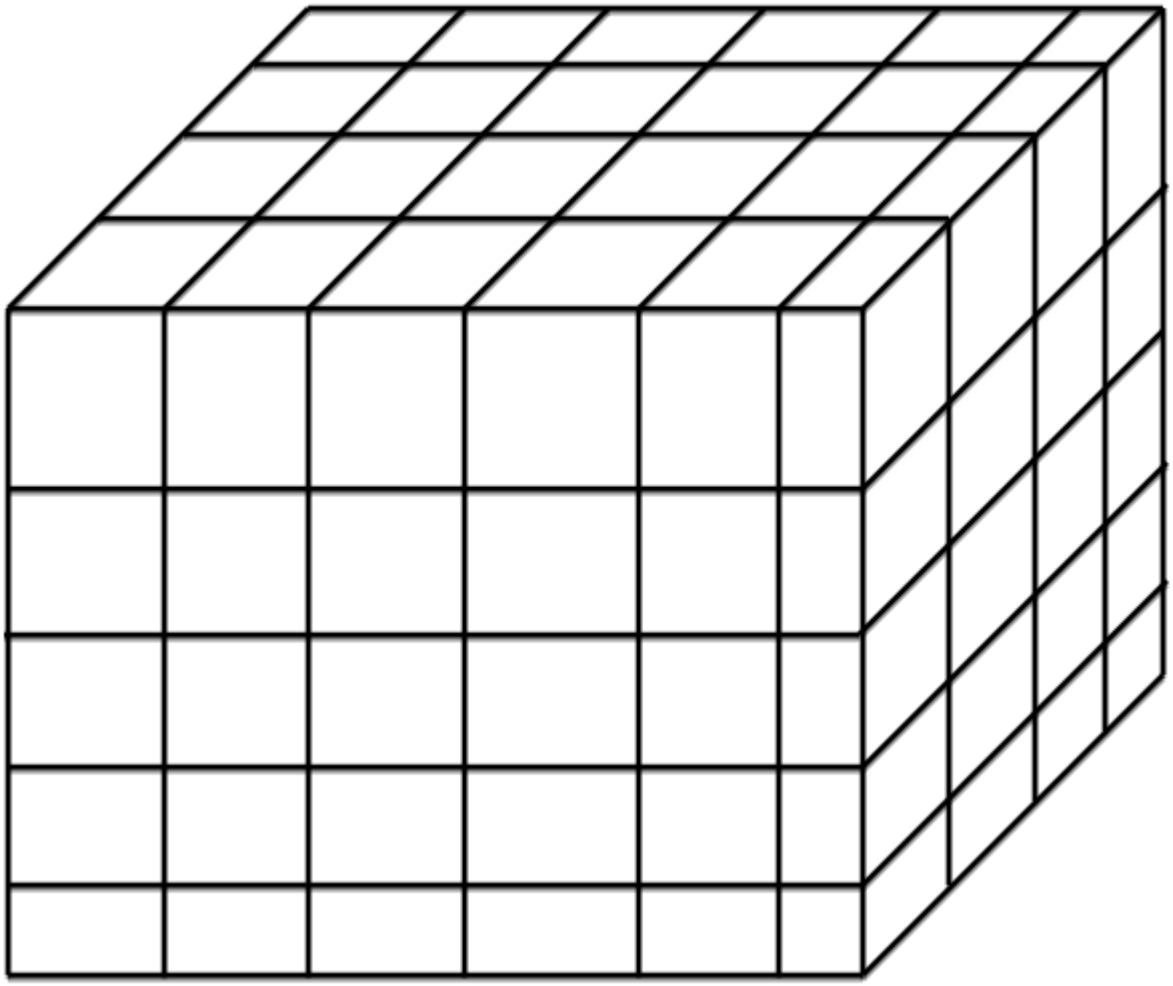
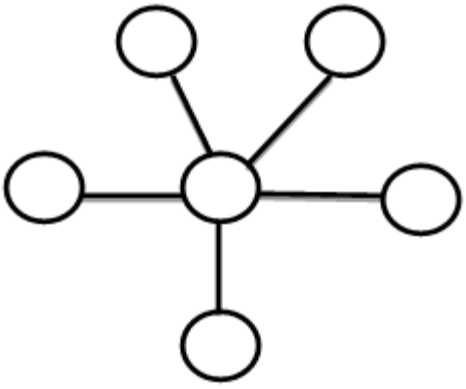


Рис. 75 CRAY T3D/T3E

2002:



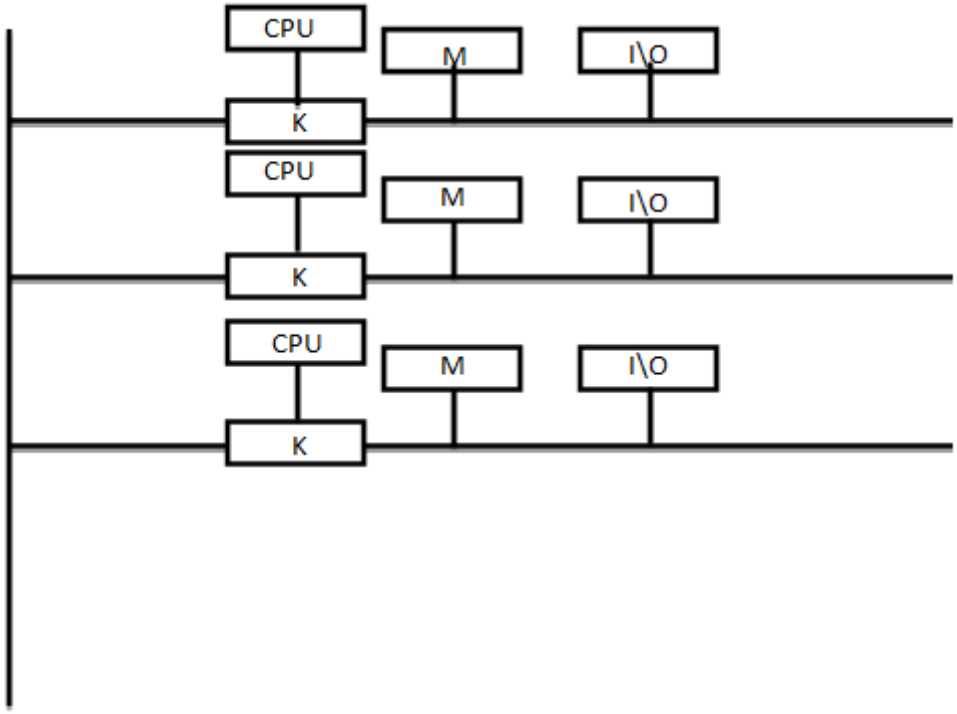


Рис. 76

1990-ті.

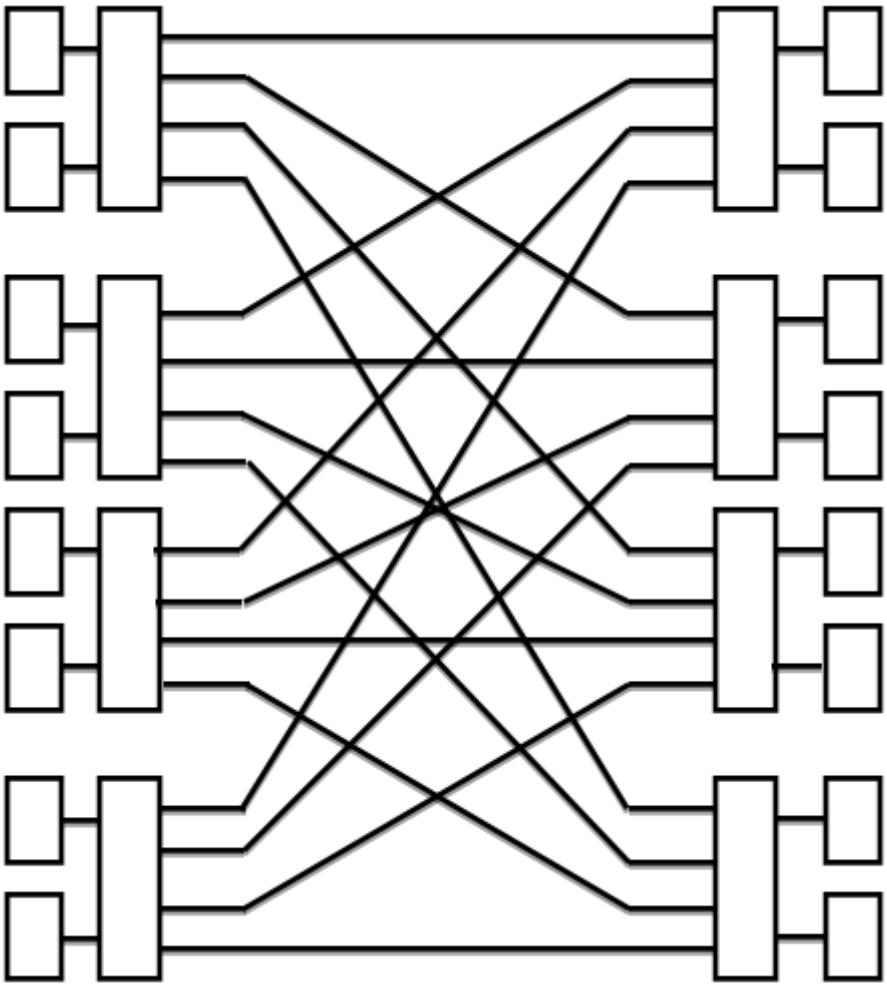


Рис. 77

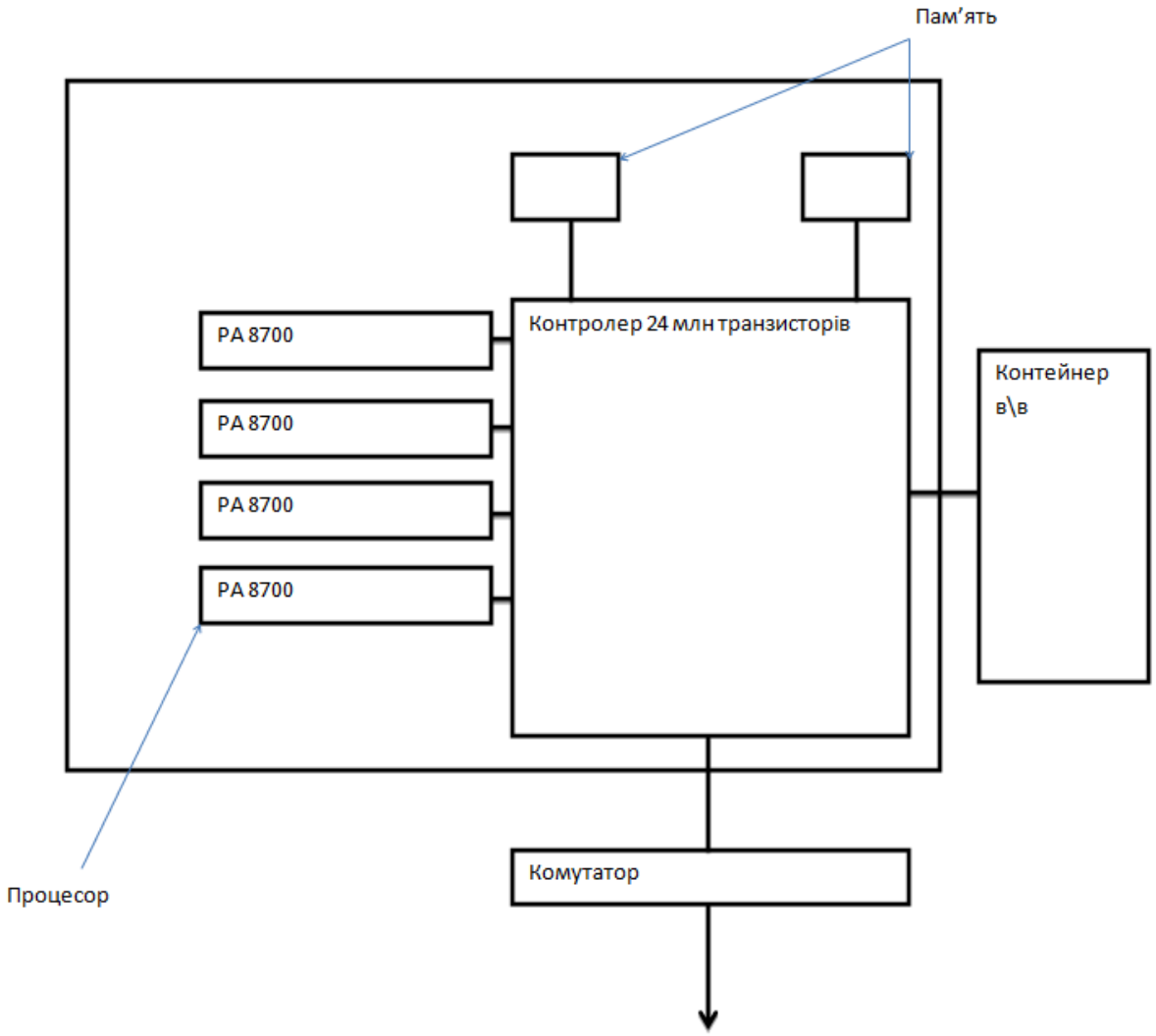


Рис. 78 HP Superdome (2006)

Один із перших кластерів The Hive (рис. 79)

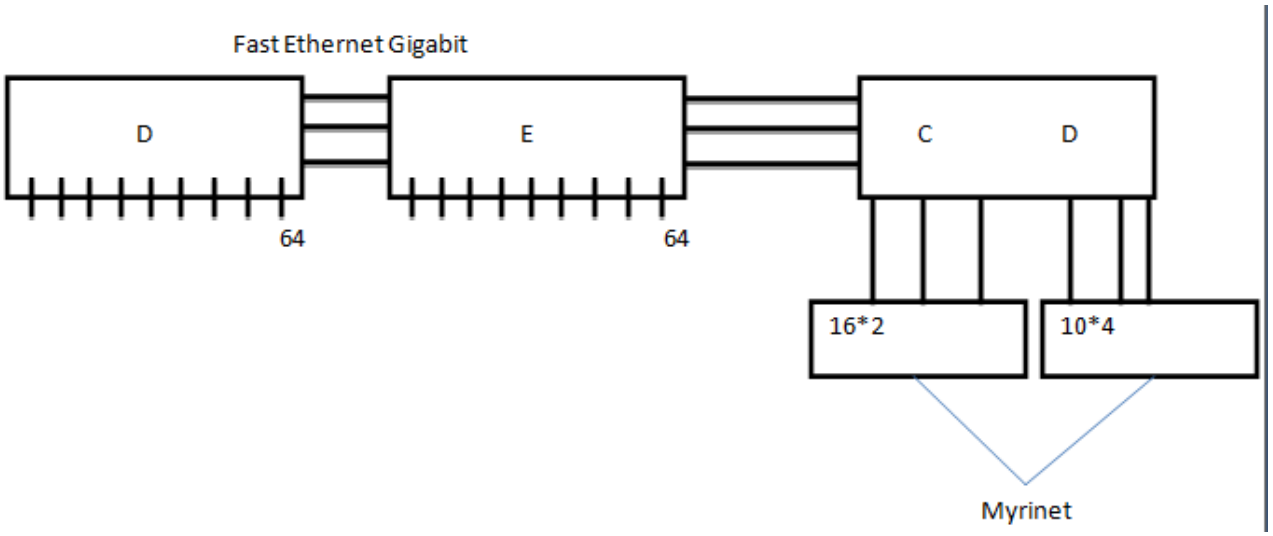


Рис. 79 The Hive

62*2+62*2+2+16*2+10*4

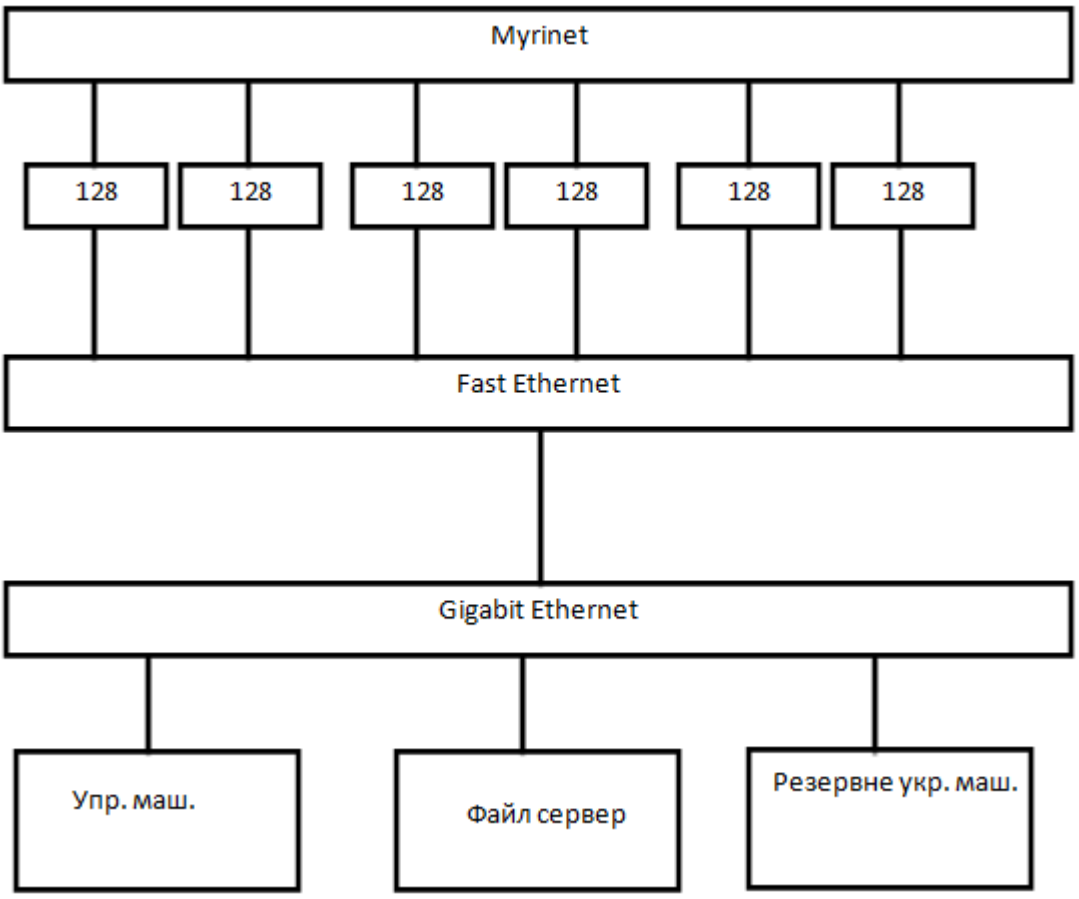


Рис. 80

GRID – Це програмно-апаратне середовище в якому ресурси або обчислювальні установки виділяються із різних адміністративних доменів телекомунікаційної мережі і які дозволяють виділяти необхідну кількість ресурсів у вигляді установок пам'яті, процесорів, програм і даних.

Таким чином, це операційне середовище, ресурси якого виділяються на основі відкритої глобальної мережі і робота, на якій повинна бути не складніша ніж робота на традиційних обчислювальних пристроях.

Основні властивості GRID:

1. Глобальне розподілення, тобто розподілення установок по глобальній мережі, або Internet, яка вимагає значних витрат на реалізацію взаємодії і представляє підвищення вимоги до безпеки даних.
2. Автономність ресурсів. Ресурси, що виділяються, ніяк не зв'язані між собою і належать різним власникам.
3. Розподіл ресурсів GRIDy пред'являється можливість колективного використання ресурсів, і тому в GRIDi повинно бути передбачено гнучке і скоординоване розподілення ресурсів між користувачами.
4. Динамічність, тобто можливість постійної і динамічної зміни кількості ресурсів і користувачів.

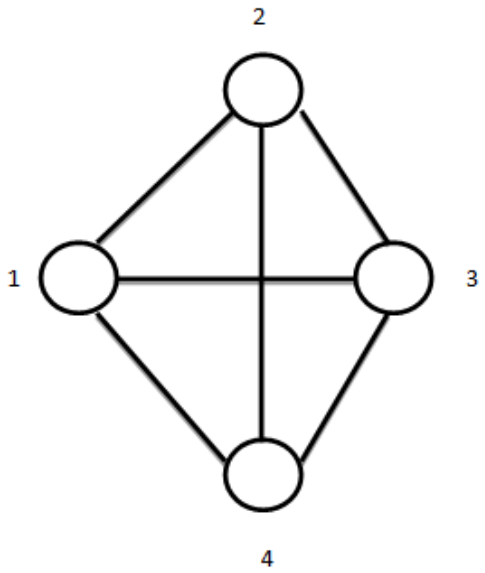
Універсальні однорідні структури з індивідуальною поведінкою елементів

Основна ідея такого середовища це:

1. Виключення фізичних зв'язків як таких. Перехід на логічні зв'язки.
2. Реалізація, принципи швидкодії в який повинен бути реалізований в максимальному варіанті, що означає безпосередній зв'язок «вихід одного елемента - вхід іншого».
3. Можливість динамічної перебудови зв'язку.

Для цього в такому середовищі кожний елемент повинен представляти собою універсальний функціонально-з’єднувальний елемент.

Для цього конфігурація середовища повинна бути такою, що б формувати її без зазорів і накладень. Досить 2 елементів крім І-НЕ:



Додаємо третій елемент:

Транспьютер

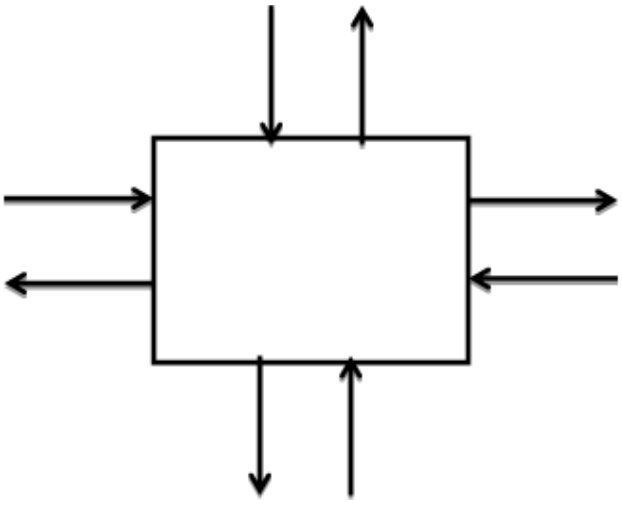


Рис. 81 Схема транспьютера

Всі входи-виходи транспьютера можуть працювати одночасно.

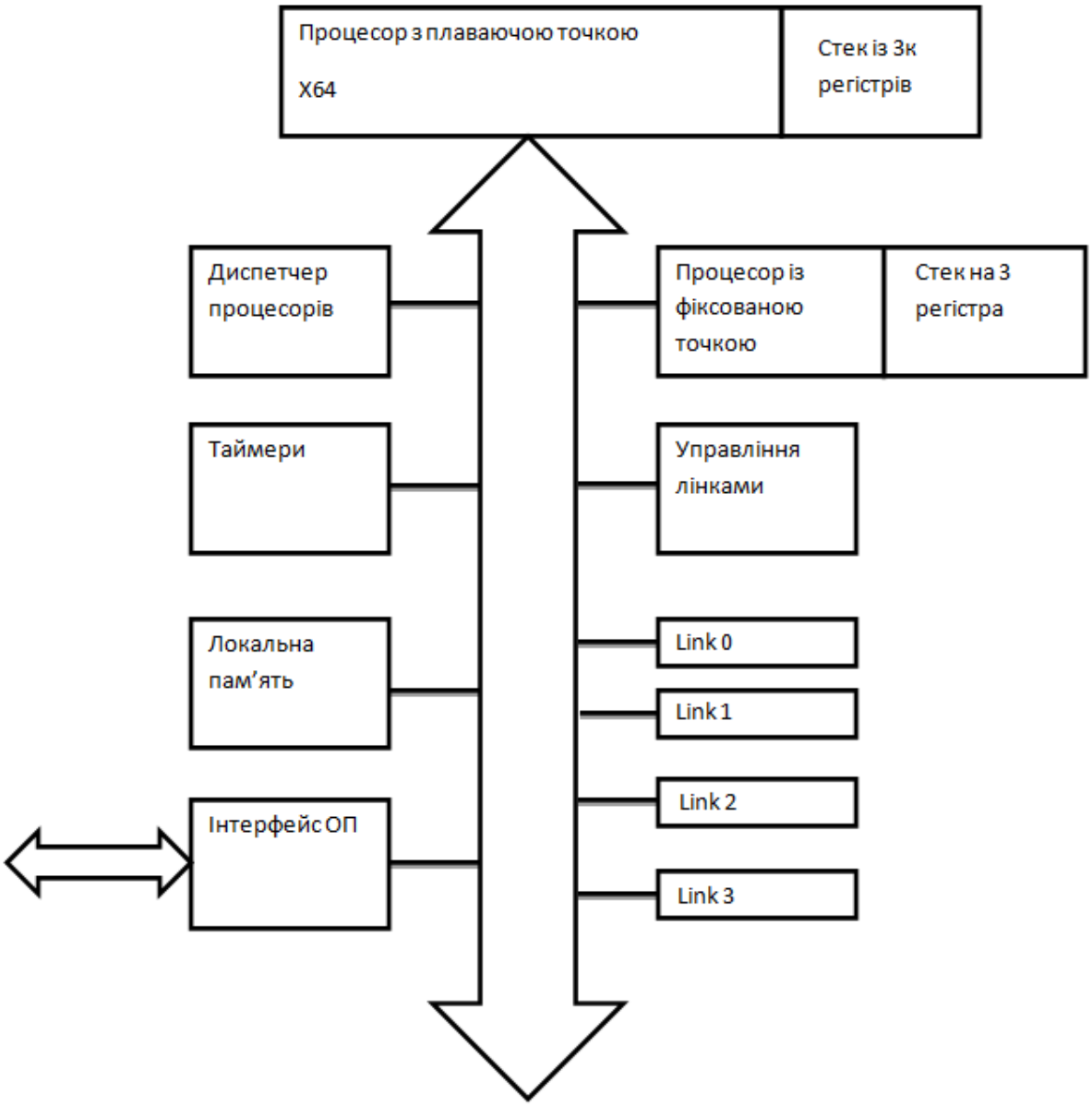


Рис. 82

Якщо канал зайнятий, то ЦП переводить в режим очікування до моменту звільнення цього каналу.

Саме признакова частина дозволяє виконувати різні дії.

В лічильник записуються розмір блока даних. Після в\в кожного елемента даних із лічильника віднімається 1(4). Це буде продовжуватись доки в лічильнику не буде 0. Завершиться виконання УСК. Після цього +1(+4) додається до А для вибірки наступного УСК.

Починаючи з адреси А будуть в\в дані по а виконується друге заняття циклу ОП для запису або зчитування наступної порції даних в регістр формувач або (він же) буферний регістр (рис. 83).

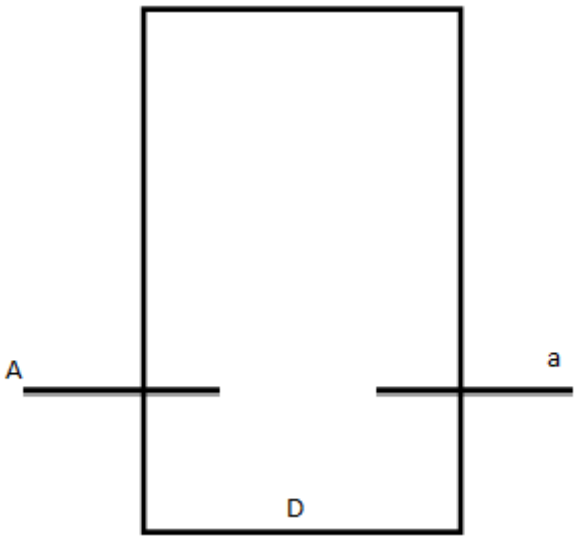


Рис. 83

Елементи класифікації вводу-виводу

Будемо розрізняти вбудовані канали в \ в, неавтономні, частково-автономні, автономні канали в \ в.

1. Вбудовані

БДОП – Блок доступу до ОП

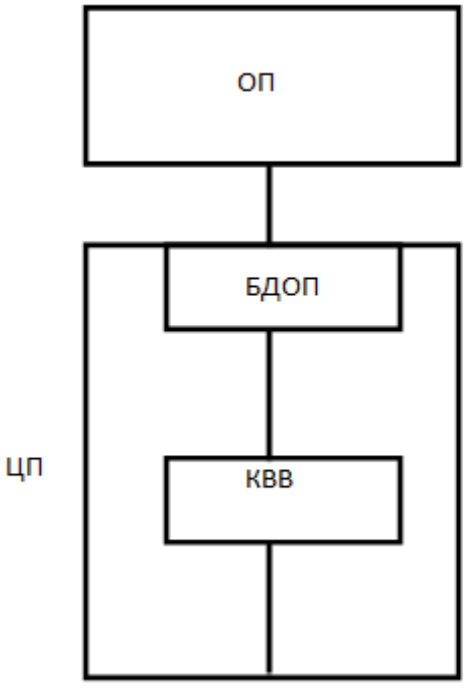


Рис. 84 Система з вбудованим КВВ

КВВ реалізований на основі ресурсів ЦП. Це той варіант, коли ЦП працює в двох режимах: рахунку і в\в.

2. Неавтономний

Рис. 85 Система із неавтономним КВВ

Неавтономний КВВ виконує ф-ли, які характерні для в\в, а саме пошук і підключення ЗП, а також контроль ЗП, а також контроль в\в. А всі пов'язані з перетворенням (А,а), зверненням до пам'яті, і т.д. залишаються за ЦП. Звідси випливає, що програмні змінні регістри ЦП використовуються для реалізації в \ в, а це означає, що має бути виконане переривання виконуються програмами при реалізації в \ в, тобто паралельне виконання прикладної програми і в \ в неможливе.

3. Частково автономний

При використанні частково автономних каналів, деякі ф-ції ЦП будуть пов'язані з в \ в але при цьому програмно змінні регістри не повинні використовуватися, що дозволяє замінити переривання призупиненням виконання прикладної програми.

а)

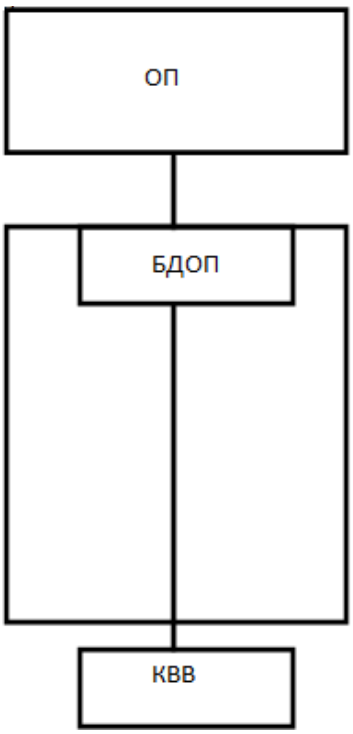


Рис. 86

б)

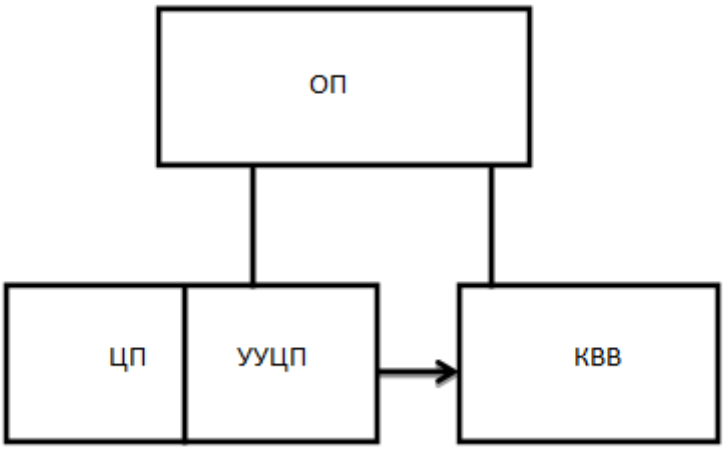


Рис. 87

Якщо деякі ф-ції у КВВ не реалізовані апаратно, він використовує мікропрограмне управління ПУ ЦП.

в)

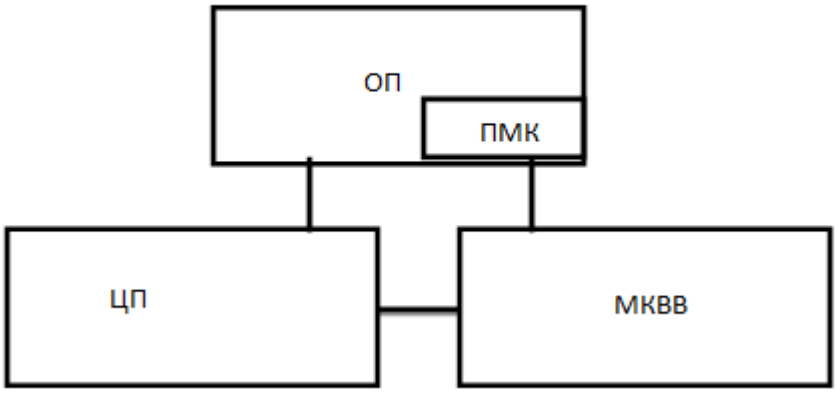


Рис. 88

ПМК – пам’ять мультиплексного каналу

МКВВ – мультиплексний КВВ

4. Автономний канал

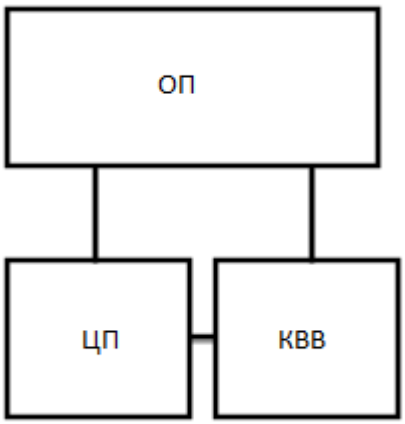


Рис. 89

При використанні автономних каналів в \ в, все таки дві тривалі процедури залишаються за ЦП:

1. Ініціювання в\в
2. Завершення в\в

Для виключення цього навантаження на ЦП, часто поряд з каналами в\в використовують процесори в\в. Процесори в\в зазвичай призначені для сумарного призначення, але відрізняються від ЦП потужністю, тобто розрядною сіткою, продуктивністю, обсягом пам'яті.

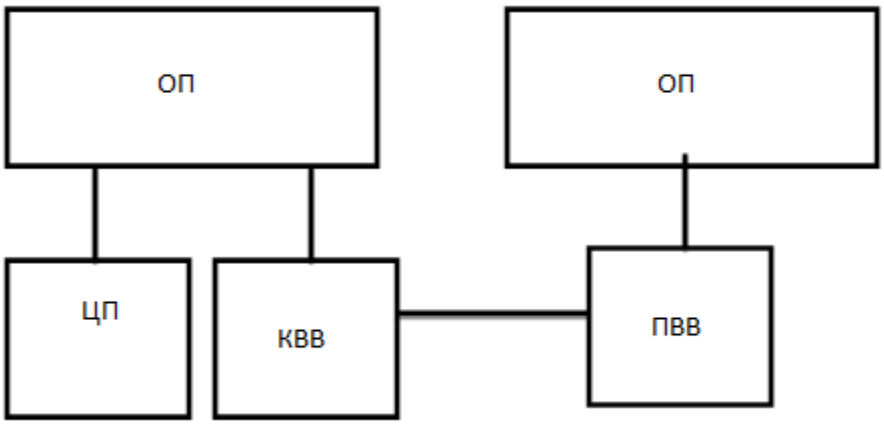


Рис. 90

Класифікація по способу підключення ЗП до КВВ

1. Канали із перехресними зв’язками
2. Канали із селективними зв’язками
3. Канали із мультиплексними зв’язками
4. Блок – мультиплексний канал

1. Канали із перехресними зв’язками

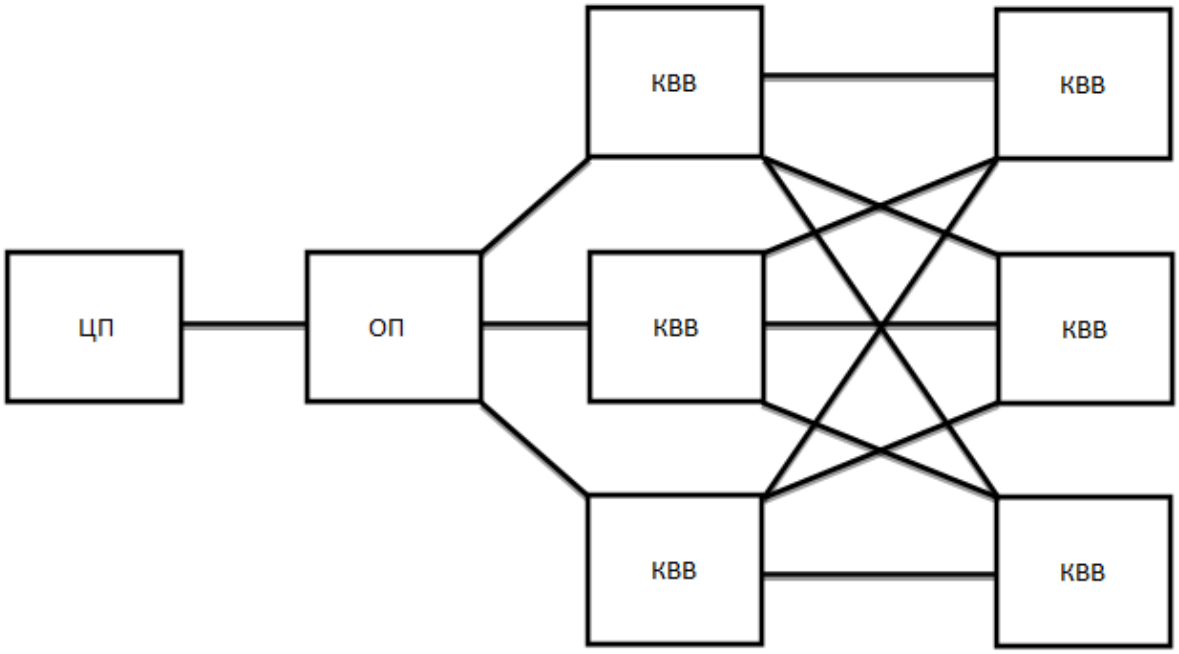


Рис. 91 Система із використанням перехресних зв’язків

Переваги:

1. Висока степінь доступності каналів для всіх ЗП

Недоліки:

1. Труднощі технічного характеру
2. Необхідність вирішення конфліктних ситуацій не тільки в прямому, але й у зворотному напрямку

2. Канали із селективними зв’язками

На відміну від 1. в селекторних каналах вся множина ЗП розділяється на ряд підмножин ВУ одного типу, і кожна така підмножина закріплюється за окремим ПВВ. Це означає що ЗП одного каналу обслуговуються строго послідовно при чому кожний ЗП підключається до каналу на весь період в\в блоку даних. Очевидно, що при такій постановці селекторним каналом можуть підключатися тільки швидкодіючі ЗП. При необхідності паралельного виконання в\в, зводах з різними ЗП, ці пристрої повинні підключатися до різних каналів.

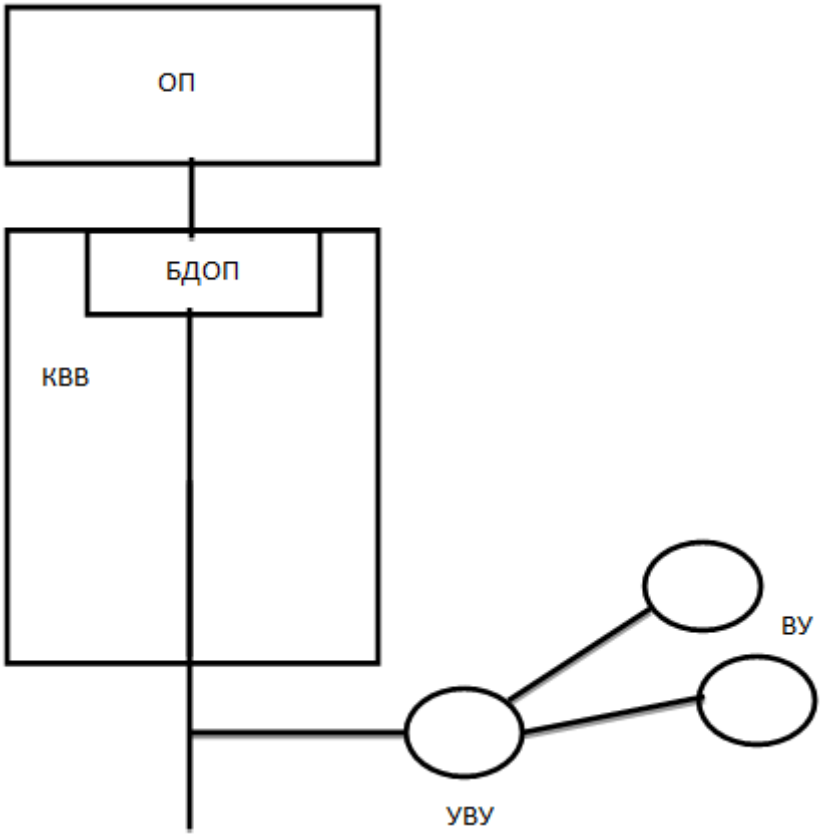


Рис. 92 Система із селекторними зв’язками

3. Канали із мультиплексними зв’язками

На відміну від селекторного каналу, і мультиплексного каналу якщо підключити повільнодіючий ЗП, оскільки такі пристрої в\в будуть відрізняться по продуктивності від КВВ на 3.4 і більше порядків, то виникає можливість обслужити всі повільнодіючі пристрої на основі 1 КВВ.

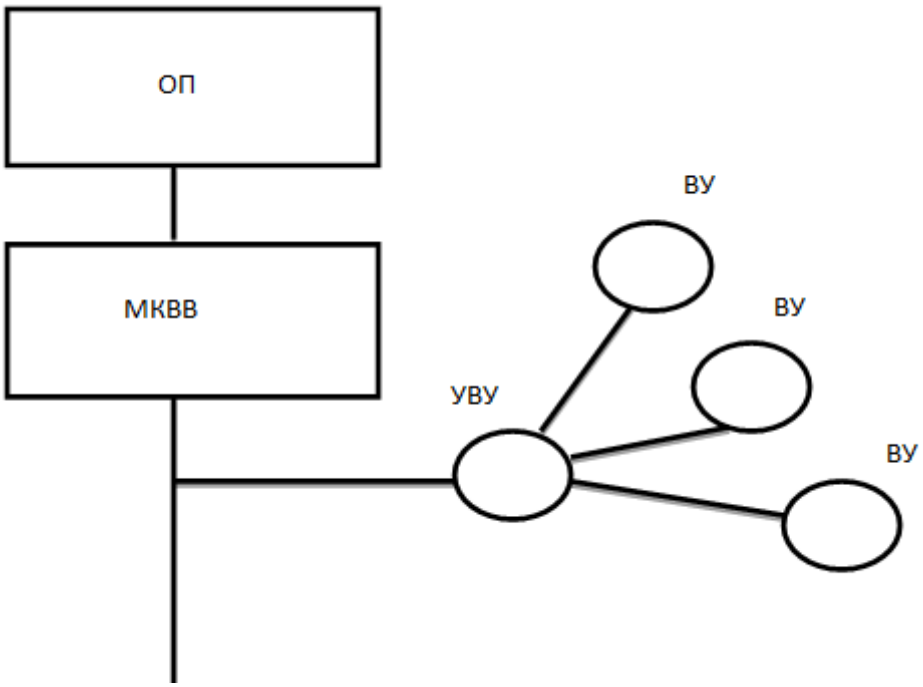


Рис. 93 Система із мультиплексними зв’язками

Мультиплексний канал працює в режимі ущільнення часу в\в, в той час як МКВВ обслуговує ЗП одного типу, готуються чергові символи на інших ЗП і кожне з них підключається до КВВ тільки тоді, коли на регістрі ЗП буде записаний черговий символ і підключення до каналу здійснюється наперед передачею цього в КВВ або навпаки (по суті виконується операція ЗП відключення від інтерфейсу в\в до підготовки чергового елемента). Після виконання операцій ЗП відключається від інтерфейсу в\в до підготовки чергового символу.

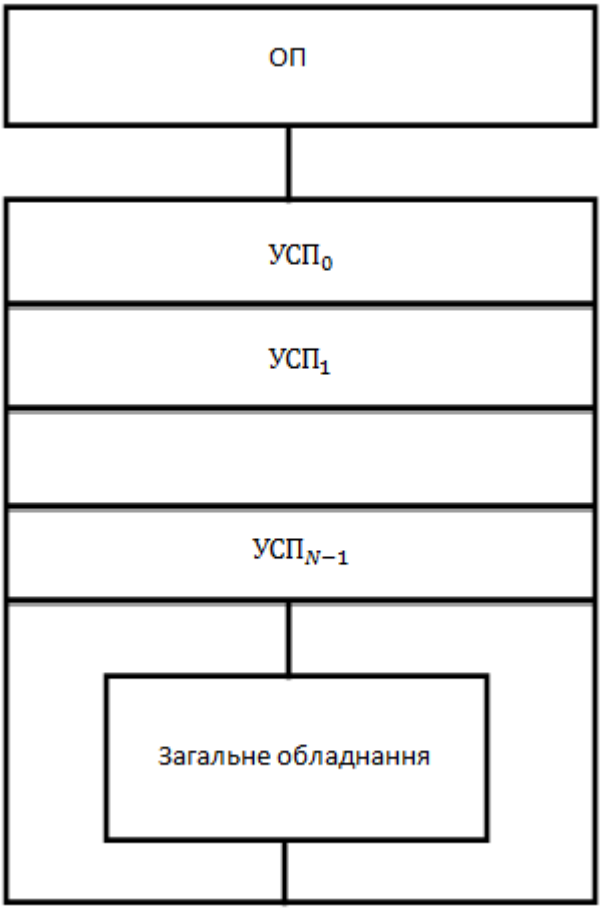


Рис. 92

УСП – Управляюче слово пристрою

ПМК – пам’ять МКВВ.

Кожне УСП зв’язане із своїм конкретним ЗП.

4. Блок- мультиплексний канал

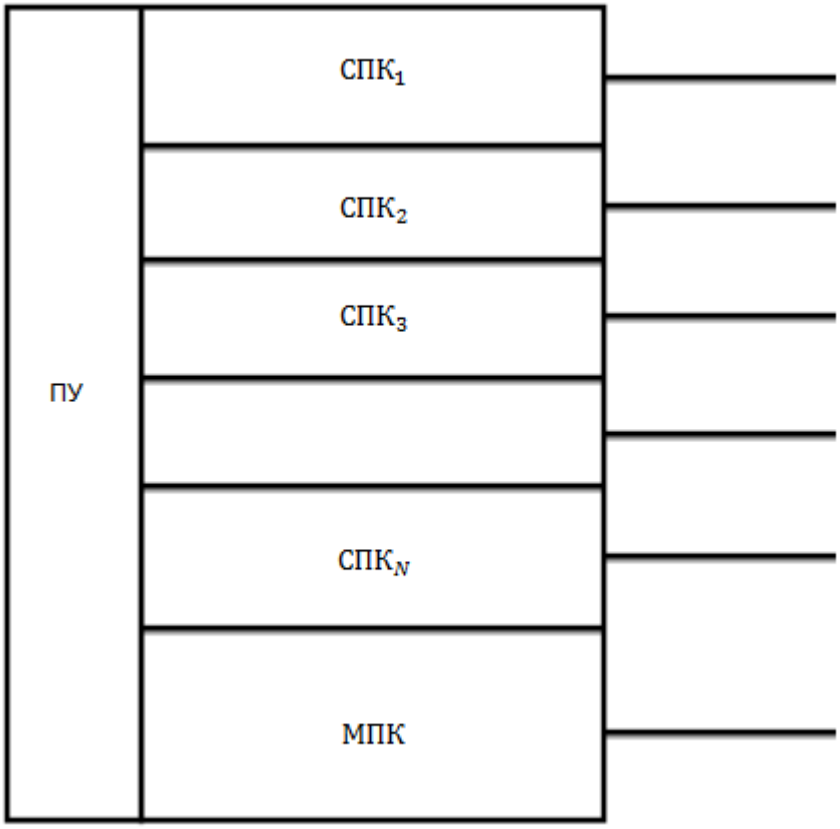


Рис. 93

МПК – Мультиплексний підканал.

$СПК_N$ – селекторний підканал

В даному випадку кожен підканал це окремий КВВ, кожен з яких володіє своїми інтерфейсами в\в. зазвичай буває 1\8 СПК і 1 МПК.

Команди управління каналами:

1. SIO – Start Input\Output – Почати ввід\вивід
2. HIO – Hold Input\Output – Зупинити ввід\вивід
3. TCH – Test Channel – Перевірити канал
4. TIO – Test Input\Output – Перевірити ввід\вивід

Це привілейовані команди, можуть виконуватися в режимі супервізора, тобто така команда в режимі користувача приведе до переривання.

SIO

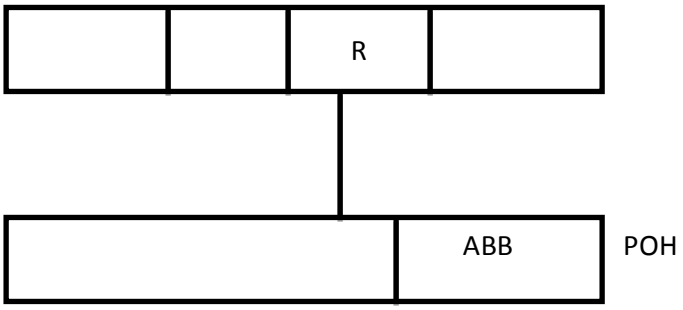
КОД	X	R	D

R – Номер індексного регістра

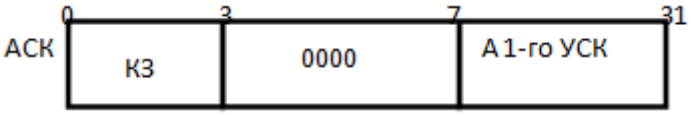
D – Зміщення

У цієї команди виконується дві операції:

1. R вказує номер індексного РОН, в якому зафіксована адреса в\в.



2. З фіксованої комірки ВП в КВВ зчитується адресне слово каналу, а точніше – АСК.

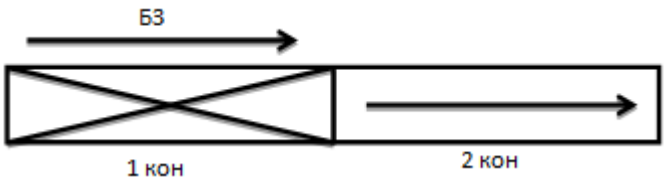


КЗ – Ключ захисту пам'яті

Ознаки:

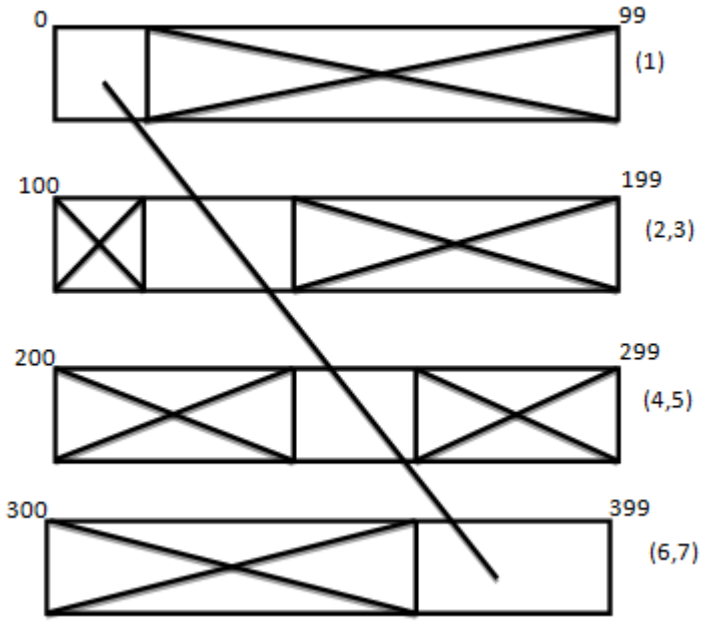
1. ЛД - ланцюжок даних, зачеплення за даними
2. ЛК - ланцюжок команд, зачеплення по командам
3. ПІНД - придушення індикації неправильної довжини
4. БЗ - блокування запису
5. ПКП - програмно кероване переривання

1. За наявності ЛД перехід до виконання наступного УСК не призводить до відключення відповідного ЗП від інтерфейсу і завершення виконання даної команди. Це означає, що 2, 3 і більше команд, зачеплених за даними сприймаються системою, як єдина команда, і швидкість переходу від одного зачеплення команд до іншого (здійснюється досить швидко), ні про яку втрату даних не йдеться.
2. ЛК. При наявності ЛК, після виконання відповідної команди здійснюється відключення ЗП від інтерфейсу і перевірка коректності виконання цієї команди.
3. ПІНД використовується в тих випадках, коли немає необхідності вводити весь блок даних, а потрібно ввести тільки частину (тобто лічильник байтів ≠ розміром блоку даних). При відсутності ознаки ПІНД це призведе до переривання, а наявність цієї ознаки виключить переривання.
4. БЗ



5. ПКП. Ця ознака може вводиться користувачем для перевірки коректності вводу-виводу в динаміці.

Приклад: ЛД і ЛК взаємовиключення



КОП	a	0	1	1	0	0		25

КОП	A+25	1	0	0	1	0		25

КОП	A+25	0	1	1	0 —	0		25

КОП	A+50	1	0	0	1	0		50

КОП	A+25	0	1	1	0	0		25

КОП	A+75	1	0	0	1	0		75

КОП	A+25	0	0	0	0	0		25

Стан вводу-виводу

D - доступний (можна використовувати)

П - зберігає переривання (не можна використовувати)

Р - працює (не можна використовувати)

Н - не працездатний

$4^3 = 64$

КВВ	УЗП	ЗП
D	D	D
D	D	П
D	D	Р
D	D	Н
D	П	Х
D	Р	Х
D	Н	Х
П	Х	Х
Р	Х	Х
Н	Х	Х

Послідовність дій при виконанні операції вводу-виводу

Щоб почати в \ в необхідно підготувати наступні елементи:

1. Програма в \ в (канальна програма)

2. АСК

3. Адреса в \ в

Якщо всі ці елементи готові, будь в \ в починається з команди SIQ. При цьому перевіряється, чи знаходиться ця команда на супервізорному рівні; Перевіряється дійсність адреси РЗП; Перевіряється стан в \ в.

Якщо всі умови для початку в \ в дотримані, тоді фіксована комірка ОП зчитує АСК, при цьому перевіряється:

1. Чи знаходиться адреса УСК на кордоні подвійного слова

2. Перевіряється дійсність адреси УСК

3. Чи є розряди 4 .. 7 нульовими

4. Чи коректна ознака по захисту пам'яті (ключ захисту)

Якщо всі умови виконані, то зчитується УСК, при цьому перевіряється:

1. Чи коректний КОП

2. Чи знаходиться адреса даних на кордоні слова

3. Чи коректна адреса даних

4. Чи не є нульовим лічильник байтів

5. Чи є нульовими розряди 37 .. 39

Якщо всі умови виконані, то І послідовності в \ в на цьому завершується.

На ІІ етапі, здійснюється зачеплення за даними і безпосередня передача даних, при цьому постійно перевіряється стан в \ в.

На ІІІ етапі, перевіряється стан ознак ЛК і ЛД. Якщо ці ознаки не рівні 0, то здійснюється перехід до І етапу, і вся ця процедура повторюється. Якщо ж ознаки ЛК і ЛД є нульовими, то перевіряється стан байта стану пристрою або байта стану каналу. Якщо один з них є нульовим, то це призводить до переривання, формування слова стану каналу і виконується завершення в \ в ЦП.

Слово стану каналу формується на основі поточних значень АСС і УСК.

32. Увага

33. УЗП зайнятий

34. УЗП закінчив роботу

35. ЗП зайнятий

36. ЗП закінчив роботу

37. Канал зайнятий

38. Помилка у каналі

39. Особливий випадок

40. ПКП

41.

42. Помилка у програмі

43. Помилка у даних

44. Помилка управління

45. Помилка інтерфейсу

46. Помилка по захисту пам’ятті

47. Помилка по захисту даних

БУС – байт уточненого стану.

Класифікація по централізації \ децентралізації

Будемо розрізняти централізовані і децентралізовані КВВ.

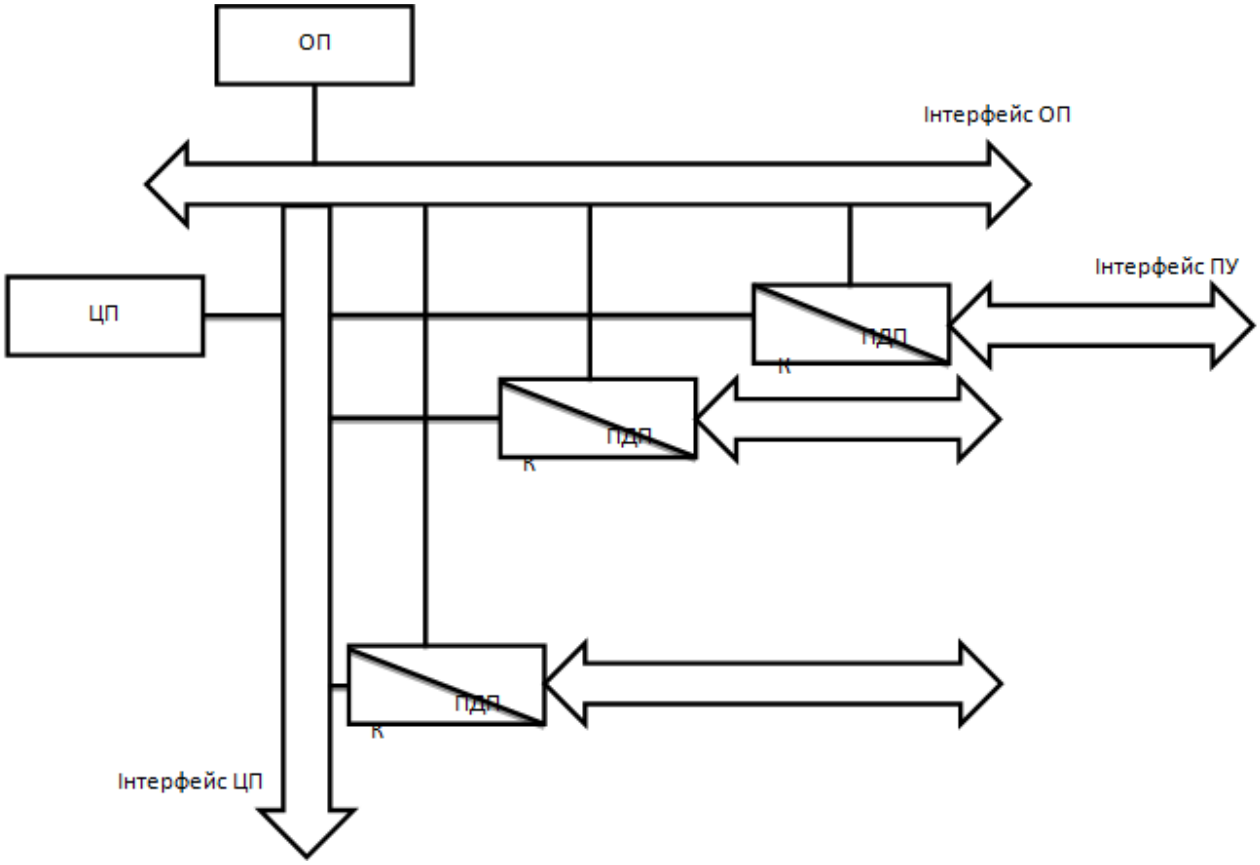


Рис. 94

Це максимально централізований спосіб системи в\в, при якому весь в\в виконаний з використанням КВВ. Однак КВВ ефективний тільки в разі введення великих блоків даних, коли операція ініціювання та довіра в\в є відносно малою у порівнянні з самою процедурою в\в.

Ініціювання та завершення в\в виконується по перериванню; виконується аналіз слова стану.

Однак, при введенні малих блоків даних, операція ініціювання та завершення можуть значно перевищувати сам в\в.

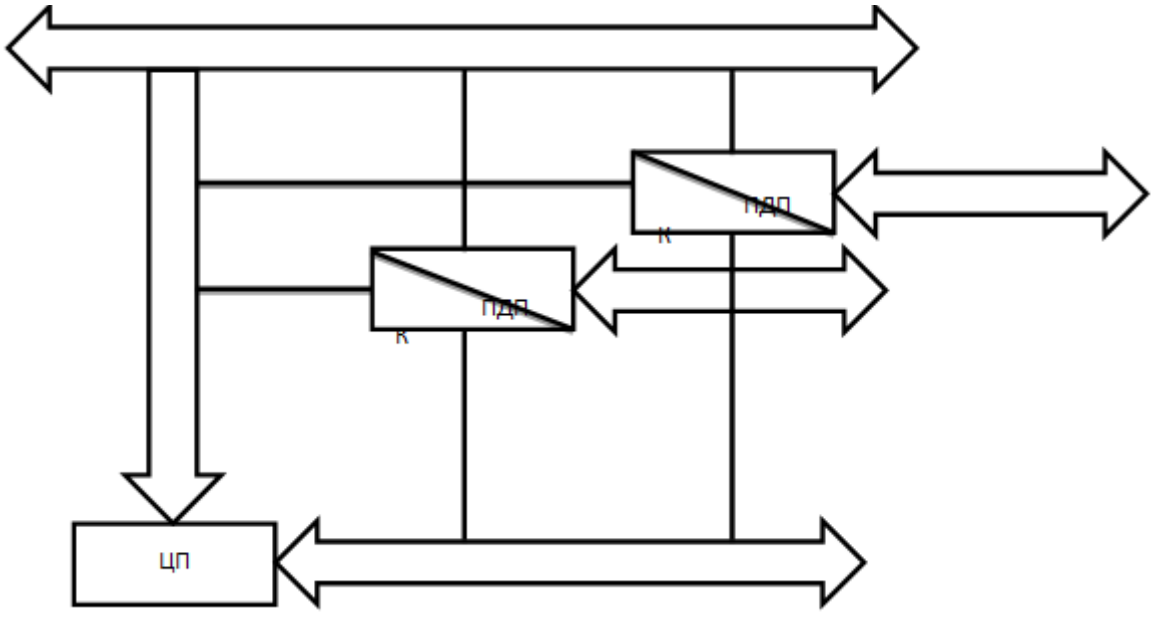
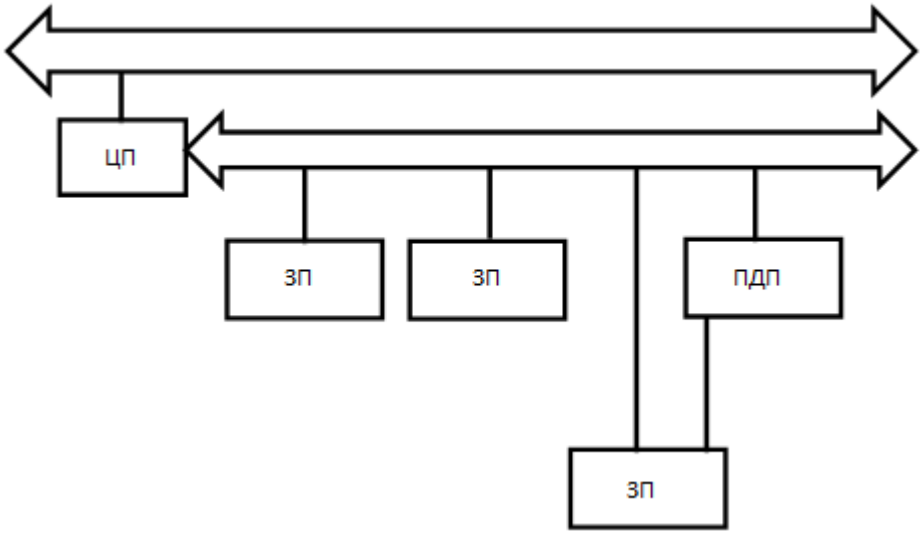


Рис. 95

Більш гнучким є другий варіант, коли для більш швидких пристроїв використовують КВВ, а для повільно діючих ЗП реалізація на основі інтерфейсу, вбудованого безпосередньо в УП.

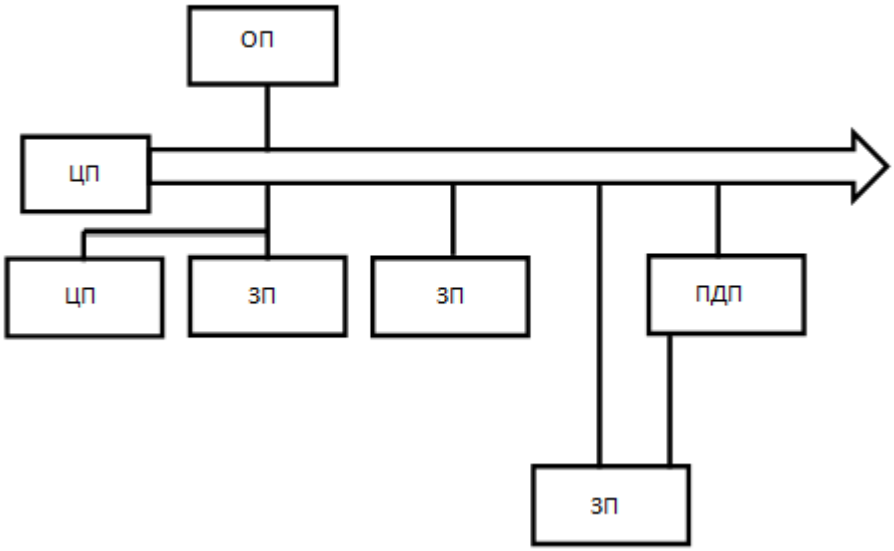
Децентралізований:

1)

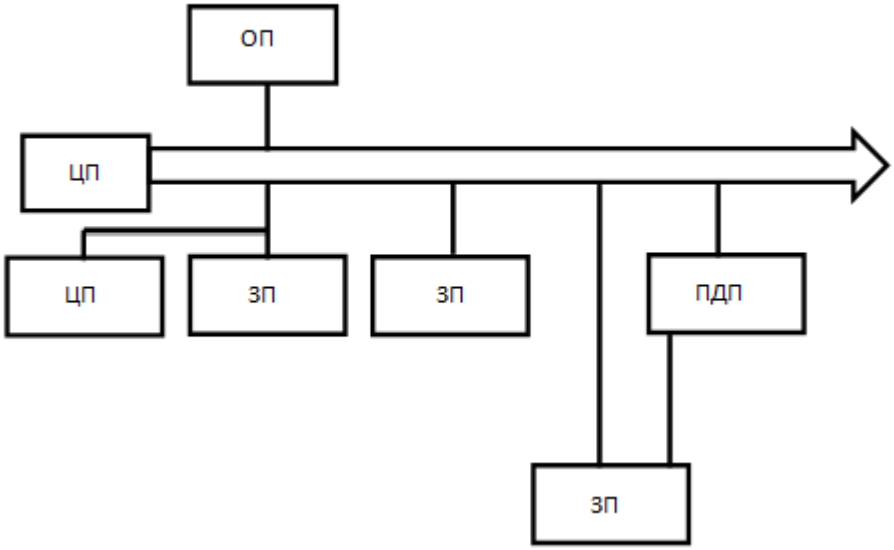


При децентралізованому способі, всі канальні функції виконуються ЦП, а зв'язок з пам'яттю здійснюється за допомогою ПДП.

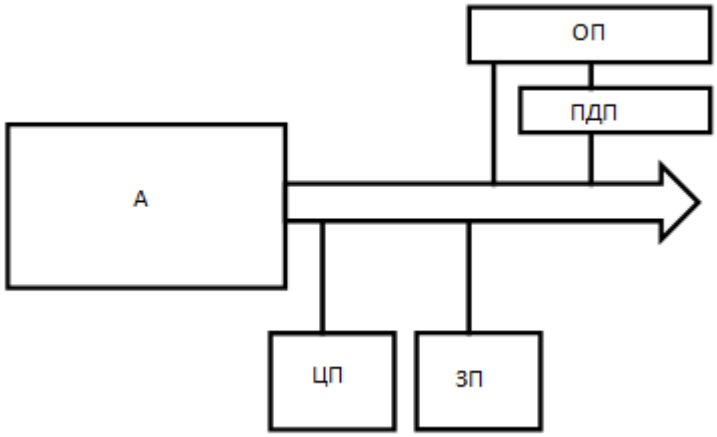
2)



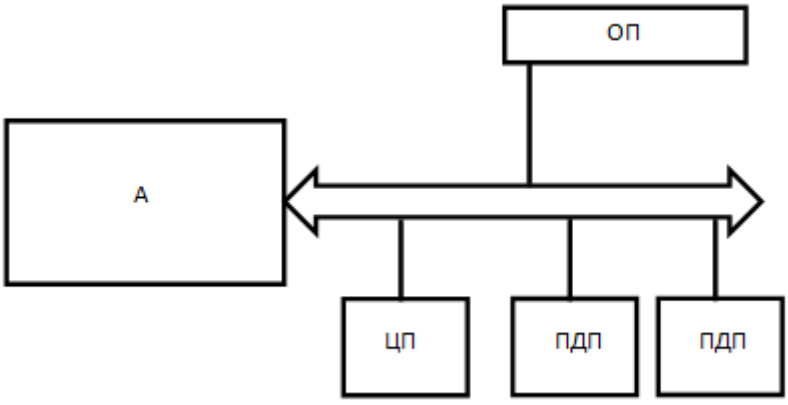
3)



4)



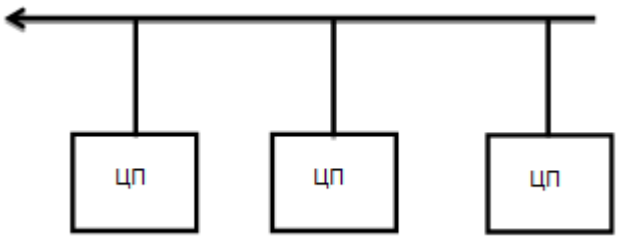
5)



Пріоритетне обслуговування переривань

При обслуговуванні в\в в разі пріоритетного їх обслуговування, будемо розрізняти:
1. програмна реалізація
2. апаратна реалізація
    1. однорівнева реалізація
    2. багаторівнева реалізація

1. Програмна однорівнева реалізація



1. Розпізнавання та обслуговування запиту

2. Ідентифікація елемента, який відправив запит

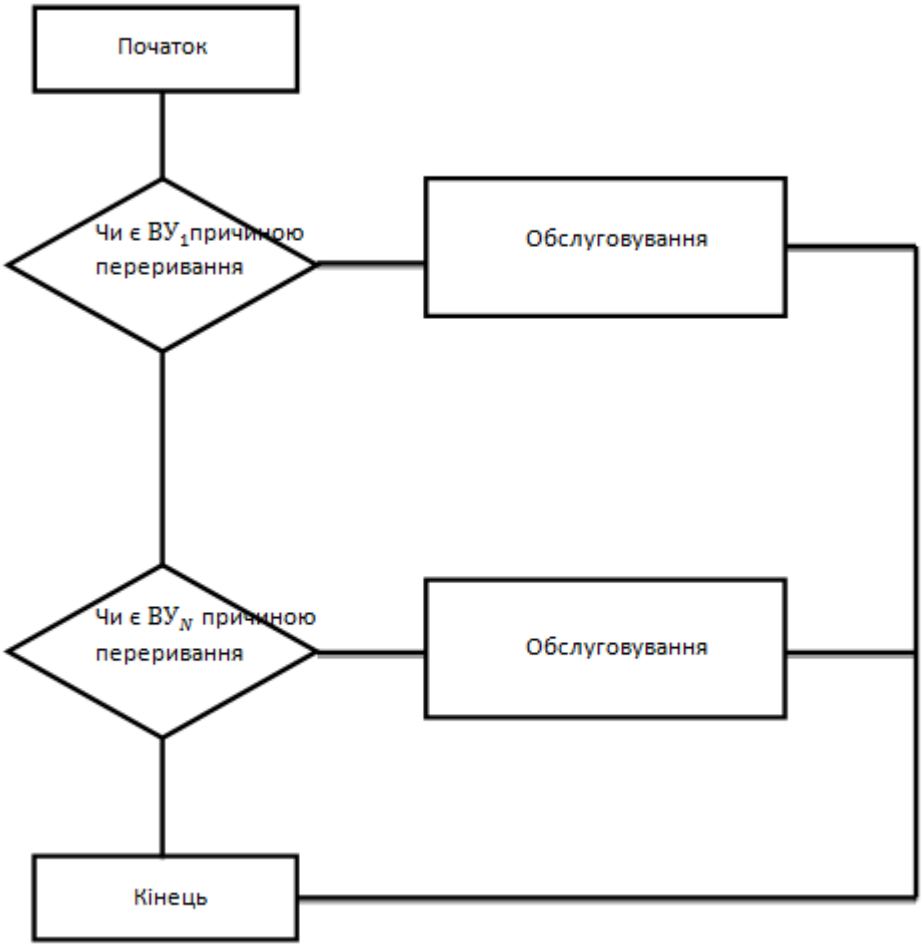
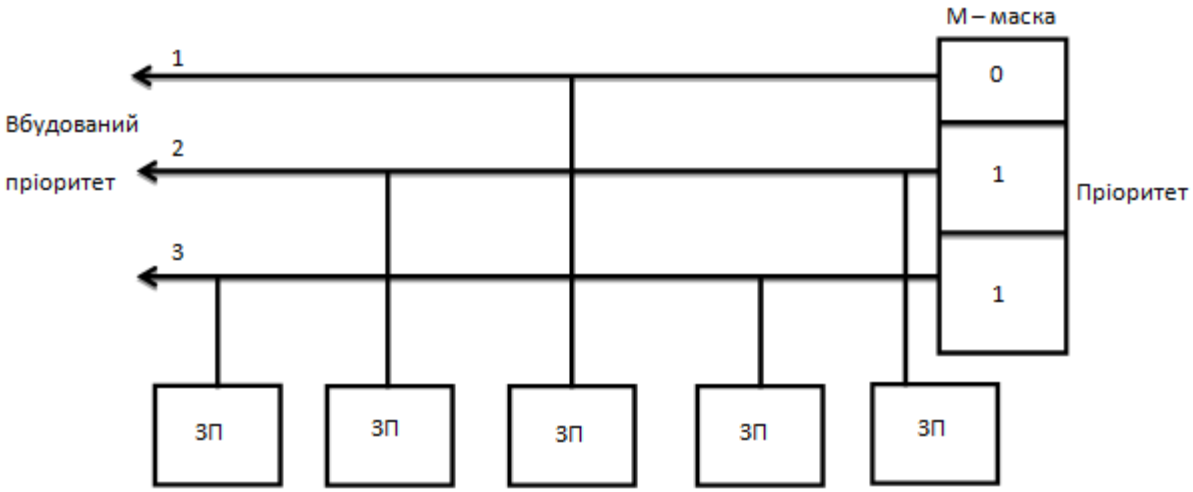
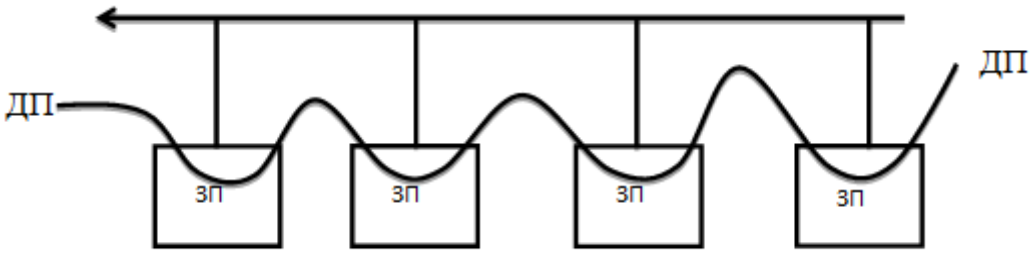


Рис. 96

2. Програмна багаторівнева організація



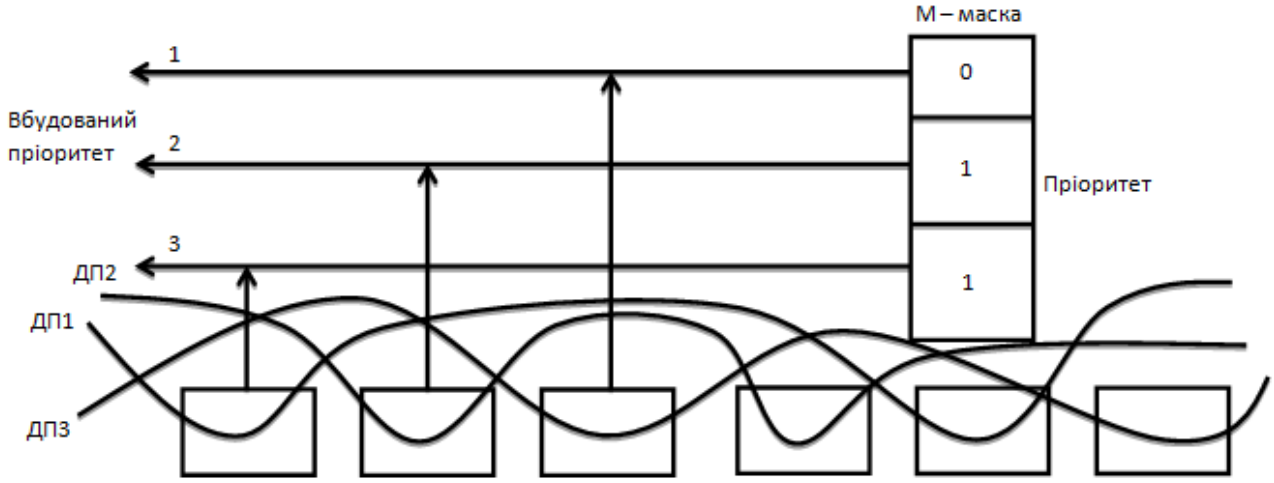
3. Апаратна однорівнева організація



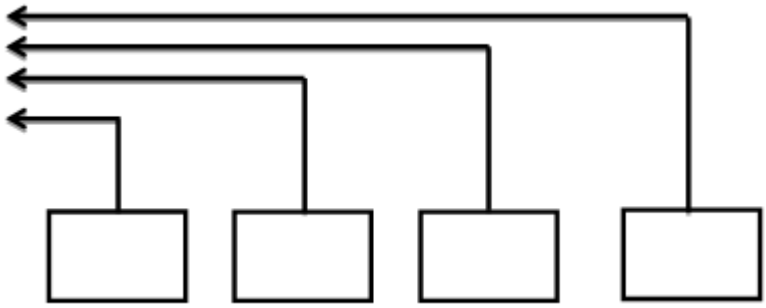
ДП – Сигнал дозволу переривання.

Жорстка система пріоритетів.

4. Апаратна багаторівнева організація



1) Виділення
2) Ідентифікація



2 в 1 – Виділення та ідентифікація.

Інтерфейси обчислювальних систем

В даний час при проектуванні систем використовується агрегатний спосіб, тобто крупно модульний варіант побудови систем, при цьому кількість модулів і їх номенклатура може визначатися користувачем, крім того в процесі ініціалізації, користувач повинен мати можливість нарощування числа модулів, модернізації старих модулів і т.д. і все це повинно здійснюється без будь-яких змін в системі, за винятком нових кабельних з'єднань нового модуля. Всі ці завдання вирішуються на основі введення стандартних засобів сполучення, які зазвичай називають інтерфейсами.

Під інтерфейсами будемо розуміти уніфіковану систему зв'язку, сигналів і алгоритмів. У високопродуктивних обчислювальних системах зазвичай використовують безліч ініціалізованих інтерфейсів, які враховують особливості модулів які вони пов'язують. Для систем не високої швидкодії може використовуватися інтерфейс типу «загальна шина».

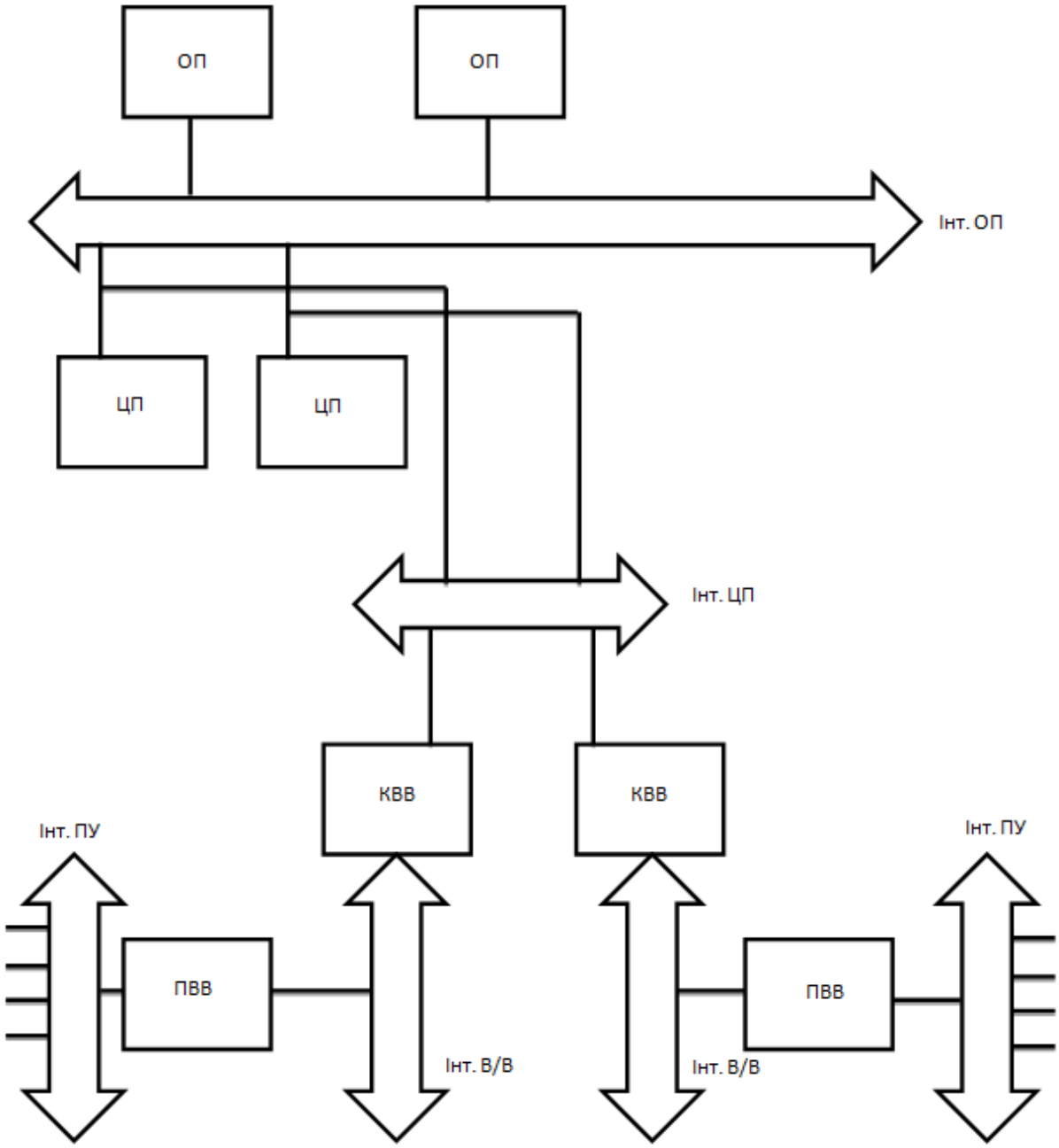
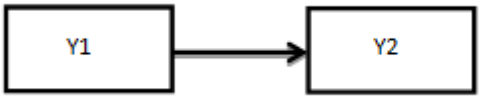


Рис. 97

Зазвичай розрізняють однозв'язні і багатозв'язні інтерфейси. У однозв'язного взаємодія між різними елементами ієрархічної системи може здійснюється тільки на основі єдиного каналу зв'язку між цими елементами. Системи з такими інтерфейсами відрізняються не високою надійністю, але найменшою вартістю, тому всі системи незначно використовують цей інтерфейс.

У разі багатозв'язних інтерфейсів між двома елементами ієрархічної системи може використовуватися 2 + каналів зв'язку. Переваги - висока надійність, недолік - висока вартість. Зазвичай ці інтерфейси використовуються в системах спеціального призначення.

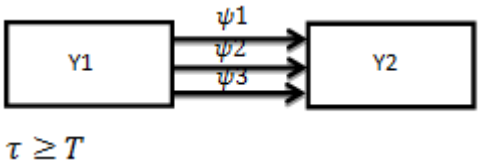
Методи передачі даних



$Y1 \rightarrow Y2$

При синхронному варіанті передачі джерело даних повинно крім передачі інформації ще й приймати підтвердження того, що приймач прийняв передану інформацію; тільки після підтвердження, він може знімати інформацію виходу.

Паралельна передача:



Присутнє явище перегонів

Сигнал не сприймається, поки не прийде останній біт.

Це метод передачі інформації зі стробуванням. Урахуванням перегонів.

Теж для асинхронного, але називається метод передачі інформації з квірів.

Організація шин інтерфейсів

Будемо розрізняти:

1. Інтерфейси з індивідуальними шинами

2. Інтерфейси з колективними шинами

3. Інтерфейси з індивідуальними і колективними шинами

При організації шин інтерфейсів будемо враховувати дві операції:

1. Адресація

2. Ідентифікація

При адресації переваги взаємодії з ПУ належить провідному пристрою (УП або каналу), звідси випливає, що ведучий пристрій на основі своєї адреси, може вирішити завдання пошуку ЗП, і його підключення до інтерфейсу.

У випадку ідентифікації, ведучий пристрій на основі сигналів від ПУ, повинно визначити його адресу і тим самим реалізувати його підключення до ведучого пристрою.

1. Інтерфейс з індивідуальними шинами

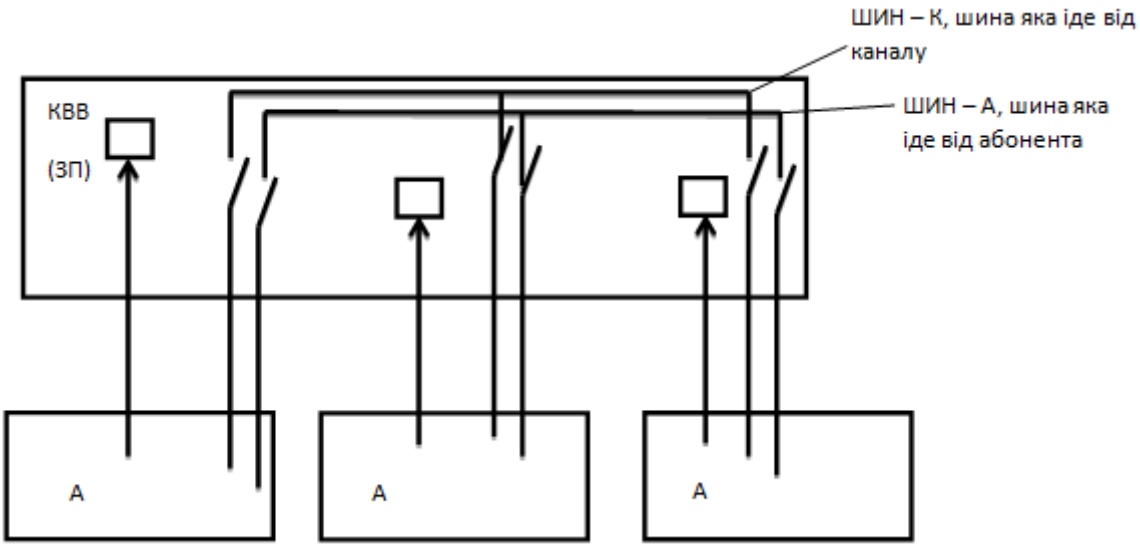


Рис. 98

Це найпростіший, але найдорожчий варіант реалізації - коли ведучий пристрій має окрему шину з кожним абонентом.

2. Поєднання колективних та індивідуальних шин

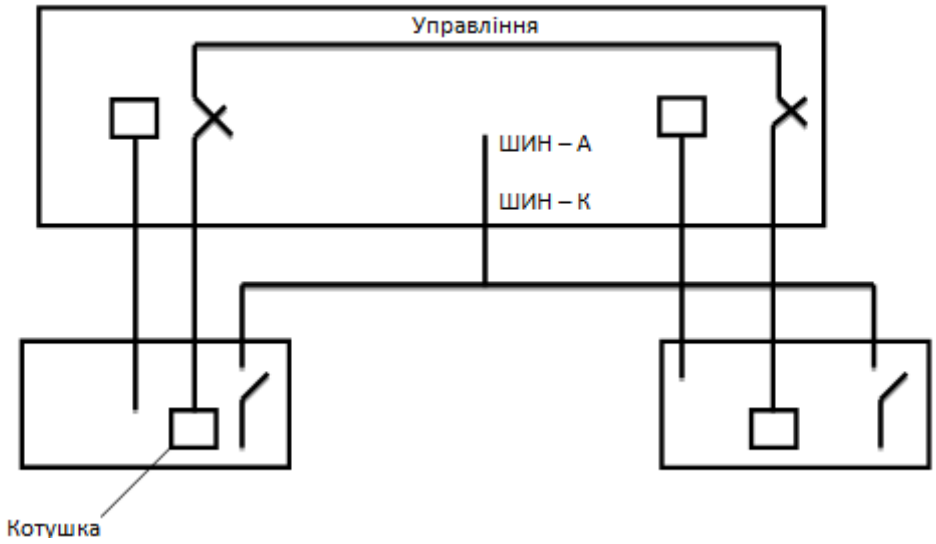


Рис. 99

3. З колективним доступом

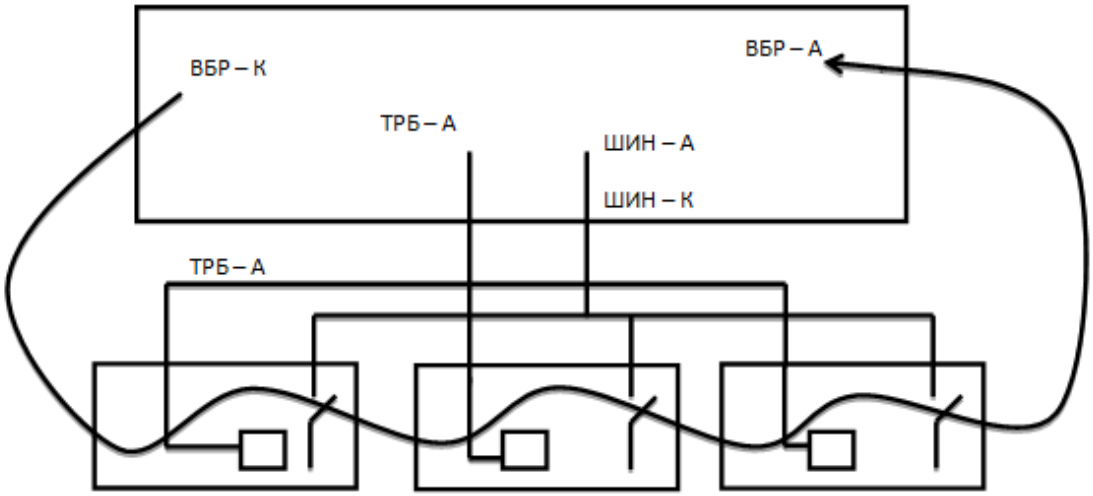


Рис. 100

Адресація. При адресації, адреса, яка йде від каналу, формується в каналі в\в і по ШИН -К передається на всі ЗП, або всім абонентам. У кожного абонента є своя індивідуальна адреса і абонент виконує операцію порівняння даної адреси і своєї індивідуальної адреси. У тих абонентів, де шини мають збіг заданої і індивідуальної адреси, здійснюється розрив лінії Вибірка від каналів і замикання на відповідну котушку. Потім на лінію ВБР -К передається відповідний сигнал, в результаті чого через задану котушку проходить струм, замикається відповідний ключ, і тим самим абонент підключається до інтерфейсу. Для перевірки коректності підключення, відповідний абонент передає свою індивідуальну адресу каналу в\в. Якщо задана адреса та індивідуальна адреса збігаються - то канал в\в передає абоненту керуюче слово, тобто задає операцію ЗП. У відповідь на це, ЗП передає в канал байт свого стану. Якщо байт стану нульовий, то канал в\в передає сигнал ПУ про те, що він починає операцію в\в. Далі йде передача даних.

34 лінії зв’язку:

ШИН – К: ШИН – 0, ШИН – 1, …, ШИН – 8; (9 ліній)

ШИН – А: ШИН – А0, ШИН – А1, …, ШИН – А8; (9 ліній)

ID: ИНФ – К, ИНФ – А, УПР – К, УПР – А, АДР – К, АДР – А; (6 ліній)

УПД: РАБ – К, РАБ – А, ВБР – К, РВБ – К; (7 ліній)

РАБ - А - Свідчить про підключення абонента до інтерфейсу, і повинен підтримуватися протягом всього періоду в\в даних абонента.

ВБР - К - За допомогою цього сигналу здійснюється підключення до інтерфейсу.

РВБ - К - Дозвіл ВБР - К, це той зв'язок, який дозволяє швидко відключаться від інтерфейсу, тобто прискорює процес звільнення каналу.

ВБР - А - Якщо ні один пристрій не вибрано, то в \ в працює не коректно.

ТРБ - А - Коли підключення здійснюється з ініціативи абонента.

БЛК - К - Канал працює не правильно, його роботу треба заблокувати.

+3 Лінії зв'язку контрольні: 9 + 9 + 6 + 7 + 3 = 34.

Управління в \ в здійснюється за допомогою наступних 5 послідовностей:

1. Початкова вибірка
2. передача даних
3. Завершення в \ в
4. Вибірка зайнятого абонента (УЗП)
5. Вибірка зініційована абонентом

1. На рівні введення \ виводу асинхронний спосіб.↑ РАБ — К
   ↑ АДР — К ↑ ШИН — К, (↑ АДР — К – передаємо адресу всім абонентам)
   Затримка
   ↑ ВБР — К ↑ РВБ — К
   ↑ РАБ — А
   ↓ АДР — К
   ↑ АДР — А ↑ ШИН — А
   ↑ УПР — К ↑ ШИН — К
   ↓ АДР — А
   ↓ УПР — К
   ↑ УПР — А ↑ ШИН — А
   ↑ ИНФ — К
   ↓ УПР — А
   ↓ ИНФ — К
2. ↑ ИНФ — А ШИН — А (якщо введення)
   ↑ ИНФ — К ШИН — К (якщо виведення)
   ↓ ИНФ — А
   ↓ ИНФ — К
3. ↑ УПР — А ↑ ШИН — А (байт стану)
   ↓ ВБР — К ↓ РВБ — К
   ↓ УПР — А ↓ РАБ — А
4. ↑ РАБ — К
   ↑ АДР — К ↑ ШИН — К
   Затримка
   ↑ ВБР — К ↑ РВБ — К
   ↑ УПР — А ↑ ШИН — А (не нульовий байт стану)
   ↓ АДР — К ↓ ВБР — К
   ↑ РВБ — К
   ↓ УПР — А
5. ↑ РАБ — К
   ↑ ТРБ — А
   ↑ ВБР — К ↑ РВБ — К
   ↑ РАБ — А
   ↑ УПР — К ↑ ШИН — К
   ↑ АДР — А ↑ ШИН — А
   ↓ УПР — К
   ↓ АДР — А