О.І. Гороховський

АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ

Міністерство освіти і науки України
Вінницький національний технічний університет

О.І. Гороховський

АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки 0915 –"Комп'ютерна інженерія". Протокол № 11 від 30.06.2005р.

Вінниця ВНТУ 2006

УДК 658.512.22
Г 77

Рецензенти:

В.М. Лисогор, доктор технічних наук, професор
В.А. Лужецький, доктор технічних наук, професор
Д.Т. Обідник, кандидат технічних наук, доцент

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Гороховський О.І.
Г 77 **АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ** . Навчальний посібник. – Вінниця: ВНТУ, 2006.- 178 с.

В посібнику розглянуто основні види забезпечення САПР для автоматизації проектування та моделювання цифрових пристроїв. Посібник розроблений відповідно до плану кафедри та програми дисципліни "Основи автоматизованого проектування засобів ОТ".

УДК 658.512.22

ЗМІСТ

ВСТУП

Мета викладання дисципліни

Підготовка спеціаліста до науково-дослідної, проектно-конструкторської, виробничо-технічної роботи, здібного розробляти і використовувати в своїй діяльності методи і засоби автоматизації проектування і виробництва засобів обчислювальної техніки.

Задачі вивчення дисципліни

Наслідками вивчення дисципліни повинні бути:

ЗНАННЯ студентами принципів побудови сучасних систем автоматизованого проектування (САПР); функціональних можливостей та особливостей організації всіх видів забезпечення САПР; загальних уявлень про сучасні вітчизняні та закордонні САПР ЕОМ; радіоелектронної апаратури (РЕА); впровадження та експлуатації САПР.

УМІННЯ використовувати САПР у різноманітних режимах при проектуванні засобів ОТ; розробляти окремі компоненти математичного та програмного забезпечення САПР; організовувати взаємодію користувача з обчислювальною системою та термінальними пристроями при експлуатації САПР.

УЯВЛЕННЯ про сфери використання та можливості сучасних САПР.

Науково-технічний прогрес породжує потреби в складніших технічних системах, які задовольняються розвитком методів і засобів як фізичної реалізації систем, так і їхнього проектування. Починаючи з 60-х років минулого сторіччя головним вираженням науково-технічного прогресу в галузі технічного проектування стала автоматизація. Роль автоматизації проектування в розвитку науки і техніки продовжує підвищуватися.

Необхідність автоматизації процесів проектування ЕОМ

Автоматизація проектування (АП) – систематичне застосування ЕОМ для виконання проектних операцій і процедур при раціональному використанні творчих здібностей людини й обчислювальних можливостей машини, покликана забезпечити принципову можливість проектування нових, надскладних систем, а при розробці об'єктів помірної складності – одержати більш високі показники, ніж при традиційному ручному проектуванні.

До *основних показників проектування* відносяться:

- якість проектованих об'єктів,
- матеріальні витрати,
- терміни проектування,
- кількість людей, зайнятих розробкою.

Автоматизація проектування виявляється в таких основних аспектах.

Перший аспект пов'язаний з автоматизацією проектування трудомістких рутинних робіт, що не вимагають творчого підходу до свого виконання (комутаційно-монтажне проектування й оформлення конструкторської документації). Формальний характер цих робіт очевидний, однак математична інтерпретація, створення алгоритмів і, отже, автоматизація цих робіт аж ніяк не тривіальні. Однак створення й удосконалювання математичного і програмного забезпечення конструкторського проектування ведеться досить інтенсивно в зв'язку з великим економічним ефектом, який одержується у результаті виконання таких робіт.

Другий аспект автоматизації проектування виражається в тому, що замість дорогого і тривалого експериментального дослідження проектованого об'єкта на макетах стали використовувати чисельні експерименти – дослідження математичної моделі об'єкта на ЕОМ. Важливо підкреслити, що *автоматизацію проектування ні в якому разі не можна розуміти як заміну ручних розрахунків машинними в рамках тих самих розрахункових методик.* Розрахунки, які орієнтовані на ручні методи, що раніше передували макетуванню, зберігаються і при автоматизованому проектуванні саме як ручні розрахунки (вони потрібні для оцінки значень параметрів у вихідних варіантах математичних моделей, реалізованих на ЕОМ). Отже, *машинні розрахунки при автоматизації проектування протиставляються не орієнтованим ручним розрахункам традиційного проектування, а заміняють експериментальне макетування.*

Третій аспект автоматизації проектування пов'язаний з розв'язанням задач, що не піддаються повній формалізації, наприклад, задач на вибір структури і принципів організації проектованих систем. Ці процедури виконуються найкваліфікованішими фахівцями. *ЕОМ при розв'язанні цих задач має допоміжне значення.* В її пам'яті зберігаються зведення про попередні розробки у даній галузі, результати вже виконаних робіт із проектування подібної системи і т.п. Цю інформацію, інженер отримує від ЕОМ у діалоговому режимі, і вона сприяє прийняттю правильних рішень у порівняно короткий термін.

До найскладніших сучасних технічних систем відносяться системи, які розроблені і виготовлені електронною і радіотехнічною промисловістю. Тому автоматизація проектування тут особливо актуальна. І не випадково, що автоматизація проектування зародилася й одержала найповніше вираження при створенні електронної й обчислювальної апаратури. На сьогодні створені й успішно функціонують декілька великих САПР, які орієнтовані на проектування різних класів радіоелектронної апаратури, на різні типи інтегральних схем і інші вироби електронної промисловості.

Проектування технічних систем це – комплекс робіт з пошуку, дослідження, розрахунків і конструювання, що мають метою одержання в прийнятій формі всіх необхідних вихідних даних для створення нових виробів чи реалізації нових процесів, що задовольняють заданим вимогам.

ЕОМ і багато інших радіоелектронних систем – складні системи, проектування яких є трудомістким процесом. Тому *виконання проектних робіт повинно бути розподілене як у часі, так і між підрозділами проектного підприємства й окремих представників колективу розробників.*

Розподіл робіт у часі призводить до поділу процесу проектування на *етапи.* Розподіл робіт між підрозділами здійснюється на основі блочно-ієрархічного підходу до проектування, що обумовлює виділення в процесі проектування ряду *рівнів.* Блочно-ієрархічний підхід дозволяє загальну складну задачу проектування об'єкта звести до сукупності простіших задач, доступних для розв'язання за допомогою наявних засобів проектування.

Автоматизація проектування виникла на базі досягнень конкретних технічних дисциплін, обчислювальної математики й обчислювальної техніки.

У конкретних технічних дисциплінах зародилися й одержали розвиток принципи побудови технічних об'єктів, прийоми і типові послідовності виконання проектних задач, системи основних понять, термінів, класифікацій, оцінок проектованих об'єктів. Багато положень, принципи і прийоми традиційного інженерного проектування сумісні з вимогами до автоматизації і вплинули на методологію сучасного АП.

Однак, при традиційному проектуванні, орієнтація на ручний розрахунок не дозволяє покласти розрахункові методи в основу виконання більшості проектних процедур. Тому *в процесі неавтоматизованого проектування переважно використовуються експериментальні методи дослідження й оцінки якості проектних рішень, одержаних на основі інженерного досвіду й інтуїції без залучення формальних методів.* З ростом складності проектованих об'єктів терміни і вартість такого проектування виявляються надмірно великими. Тому виникла необхідність у переході від фізичного експериментування до математичного моделювання, у заміні евристичних прийомів оцінки, у визначенні параметрів і оформленні документації алгоритмізованими процедурами.

Обчислювальна математика дала можливість алгоритмізувати і автоматизувати ряд проектних процедур, що мають відому математичну інтерпретацію.

Однак математична постановка для більшості проектних процедур неочевидна, а їх подальша алгоритмічна реалізація існуючими математичними методами часто незадовільна. Тому *формалізація задач,*

вибір і розробка математичних моделей, методів і алгоритмів виконання проектних процедур значною мірою визначають зміст теорії АП.

Необхідна умова реалізації алгоритмізованих проектних процедур – наявність відповідних засобів обчислювальної техніки. Особливістю АП є існування задач, можливості розв'язання яких без прийняття припущень, що спрощують, перебувають за межами можливостей обчислювальної техніки, як уже створеної, так і передбаченої в найближчому майбутньому. Тому у великих САПР знаходять застосування ЕОМ дуже високої продуктивності. Пристосування задач до можливостей наявної обчислювальної техніки відбувається, по-перше, на основі спеціальних прийомів поділу процесів проектування на ряд ієрархічних рівнів і аспектів, по-друге, завдяки збереженню за людиною в САПР тих функцій, що не можуть бути виконані формальними методами з прийнятними витратами часу і засобів. В результаті *процес автоматизованого проектування зводиться до необхідності розв'язання кінцевої послідовності задач прийнятної складності в режимі взаємодії людини й ЕОМ.*

Необхідність взаємодії людини й ЕОМ і специфіка проектних задач породжують багато додаткових вимог до технічних засобів. Крім пристроїв програмної обробки даних необхідні спеціальні пристрої оперативного обміну інформацією, документування й архівування проектних рішень, збереження інформації у вигляді бази знань, що відбиває накопичений досвід проектування. Засоби взаємодії людини з ЕОМ повинні бути багаторазово дубльовані і наближені до робочих місць інженерів. Питання організації спільного функціонування різноманітних і чисельних технічних засобів у складі САПР складають важливий розділ теорії АП. Ці питання стосуються не тільки апаратних засобів, але навіть, у більшій мірі, засобів програмного забезпечення САПР.

Таким чином, *автоматизація проектування як науково-технічна дисципліна містить у собі*:

- методологію АП;
- математичне забезпечення, що поєднує математичні моделі, методи й алгоритми для виконання різних проектних процедур;
- питання комплексності технічних засобів і розробки спеціалізованої апаратури для САПР;
- питання розробки і використання програмно-інформаційного забезпечення банків даних, пакетів прикладних програм, операційних систем ЕОМ.

Передумови та можливості автоматизації проектування ЕОМ

Застосування ЕОМ для розв'язання інженерних задач почалося відразу ж після появи перших ЕОМ. Однак це застосування жадало від користувача трудомісткої підготовки задач до розв'язання, що полягає в

математичному формулюванні задачі, виборі чисельного методу, розробці алгоритму і його запису однією з мов програмування. *Автоматизоване проектування відрізняється від подібного використання ЕОМ насамперед тим, що майже усі з перерахованих операцій автоматизовані і виконуються на ЕОМ за допомогою заздалегідь розробленого програмного забезпечення, розрахованого на багаторазове застосування при розв'язанні визначеного класу проектних задач.*

Від користувача потрібно лише описати вихідні дані задачі проблемно-орієнтованою мовою і бути готовим до оцінки результатів і прийняття рішень за отриманими від ЕОМ розрахункам. Перші програмні комплекси, що забезпечують роботу інженера з ЕОМ створені на початку 60-х років минулого сторіччя для потреб проектування в радіоелектроніці, електронній техніці, будівельній механіці, літакобудуванні. Однак ступінь автоматизації проектування за допомогою окремих комплексів подібного типу залишалася невисокою. За допомогою ЕОМ виконувалася тільки частина необхідних проектних процедур. Були відсутні засоби підтримки взаємодії комплексів між собою. У результаті переходи від однієї процедури до іншої вимагали трудомісткого ручного інформаційного узгодження даних. Але навіть при такому недосконалому використанні засобів АП у ряді випадків вдалося підвищити ефективність проектування (скоротити терміни), поліпшити його якість, знизити матеріальні витрати, ліквідувати потреби в збільшенні чисельності колективів проектувальників.

Однак *істотне підвищення ефективності спостерігається тільки при наскрізній автоматизації проектування за допомогою комплексу засобів, об'єднаних в єдину систему автоматизації проектування.* В наш час створені і функціонують великі САПР у радіоелектронній і машинобудівній промисловості, випускаються програмно-апаратні комплекси (інтерактивні графічні станції), що можуть використовуватися як автономно, так і в складі обчислювальних мереж САПР. Такі станції широко застосовуються на багатьох підприємствах. Спостерігається тенденція до інтеграції автоматизованих систем проектування у вигляді систем *гнучких автоматизованих виробництв (ГАВ).* Застосування ГАВ дозволяє істотно підвищити ефективність виробництва шляхом швидкої перебудови обладнання на випуск визначених класів деталей. САПР у складі ГАВ призначена для оперативного проектування технологічних процесів, оснащення й інструмента, одержання результатів проектування виробів у вигляді інформації на машинних носіях для керування технологічним обладнанням.

Технічні засоби (ТЗ) і загальне системне програмне забезпечення (ПЗ) є інструментальною базою САПР. Вони утворюють фізичне середовище, в якому реалізуються інші види забезпечення САПР (математичне, лінгвістичне, інформаційне й ін.). Інженер, взаємодіючи з

цим середовищем і вирішуючи різні задачі проектування, здійснює автоматизоване проектування технічних об'єктів. Технічні засоби і загальне програмне забезпечення в процесі проектування виконують різні, але взаємозалежні функції із забезпечення перетворення інформації і передачі її в просторі і часі.

Задачі, які розв'язуються на різних етапах проектування, можна розділити на три групи:

- синтез;
- аналіз;
- тестування (перевірка).

Задачі синтезу спрямовані на побудову процедур *структурного* і *параметричного* синтезу:

- При *структурному* синтезі визначається структура об'єкта – безліч складових його елементів і способи їхнього зв'язку між собою (у складі об'єкта і з зовнішнім середовищем).
- *Параметричний* синтез полягає у визначенні числових значень параметрів елементів при заданій структурі й умовах працездатності (тобто необхідно знайти зону у просторі внутрішніх параметрів, в якій виконуються ті чи інші умови).

Аналіз полягає у визначенні властивостей і дослідженні працездатності об'єкта за його описом. Задачі аналізу спрямовані на побудову процедур моделювання і верифікації. При *моделюванні* досліджується поведінка проектованого об'єкта на моделі – програмі, яка обчислює сукупність необхідних математичних співвідношень при заданих вхідних впливах. *Верифікація* націлена на доказ правильності проектних рішень (розроблених технічних і програмних засобів).

При виробництві сучасних засобів обчислювальної техніки велике значення мають контроль і діагностика. Розв'язання цих задач дозволяє прискорити налагодження електронного обладнання, виявлення несправностей при збоях, гарантує правильність його роботи і підвищує надійність. У процесі *контролю* виявляється факт справної чи несправної роботи, а при *діагностиці* автоматично знаходяться компоненти, що відмовили. Усе більша увага приділяється проблемам самодіагностики і саморемонту, коли електронні блоки включають власні схеми виявлення несправностей і засоби їхнього усунення.

Автоматизація проектування припускає часткове чи повне розв'язання перерахованих вище задач у рамках розглянутих етапів на універсальних чи спеціалізованих ЕОМ. Можливості автоматизованого проектування складних технічних об'єктів обумовлені використанням ряду принципів, головними з який є *декомпозиція* і *ієрархічність* описів, *етапність* і *ітераційність*, *типізація* й *уніфікація*. Загальний процес проектування розділений на окремі етапи. На кожному з них використовується свій рівень абстрагування. Наприклад, на структурно-

логічному етапі розв'язуються задачі синтезу, аналізу і тестування структурних блоків ЕОМ (процесора, систем введення-виведення і т. д.). При цьому значно менший інтерес становлять конструкторські проблеми. Звичайно, між різними етапами є певний взаємозв'язок.

Принципи декомпозиції, ієрархічності і ітераційності поширюються в глиб етапів і приводять до появи ієрархічних рівнів. Таким чином, *перетворення технічного завдання в остаточний продукт породжує проміжні описи об'єкта проектування на різних етапах і рівнях, які називаються проектними рішеннями.* Якщо розв'язання задач вищих ієрархічних рівнів передує розв'язанню задач нижчих ієрархічних рівнів, то проектування називають *спадним (СП)*, у іншому випадку – *висхідним (ВП)*. На практиці їх звичайно комбінують. Наприклад, при проектуванні засобів, в яких використовуються уніфіковані елементи, розробка яких здійснюється раніш, ніж об'єктів на їхній основі.

Послідовність проектних процедур, використовувана для проектування заданого об'єкта, називається ***маршрутом проектування.***

За характером і ступенем участі людини й ЕОМ у виконанні деякого маршруту розрізняють такі режими проектування:

- *автоматичний*, що має місце при виконанні маршруту проектування за допомогою ЕОМ без втручання людини в хід проектування;
- *ручний*, що характеризується виконанням маршруту без допомоги ЕОМ;
- *автоматизований*, при якому частина проектних процедур виконується людиною вручну, а частина – за формальними алгоритмами на ЕОМ;
- *діалоговий* (інтерактивний), при якому всі процедури в маршруті виконуються за допомогою ЕОМ, а участь людини виявляється в оперативному оцінюванні їхніх результатів (у виборі продовжень і корегуванні ходу проектування). Якщо ініціатором діалогу є людина, якій надана можливість у будь-який момент перервати автоматичні обчислення на ЕОМ, то діалог називається *активним*. Якщо переривання обчислень відбуваються за командами виконуваної на ЕОМ програми в певні, заздалегідь передбачені моменти, тобто проектувальник не може виступати як ініціатор, то такий діалог називається *пасивним*.

Розробка САПР – це велика науково-технічна проблема; впровадження САПР вимагає значних капіталовкладень. Досвід розробки САПР, створених на сьогодні, дозволяє виділити такі основні концепції:

1. Взаємодія людини й ЕОМ у процесі проектування.
2. Реалізація принципів етапності, декомпозиції і ієрархічності (блочно-ієрархічного підходу).

3. Подання САПР у вигляді ієрархії інформаційно-узгоджених підсистем. Інформаційна узгодженість означає, що послідовно йдуть проектні процедури в маршрутах проектування реалізуються інформаційно-узгодженими програмами. Дві програми мають цю властивість, якщо вихідні дані однієї з них можуть бути вхідними для іншої і при цьому не потрібно ніяких перетворень.
4. САПР повинна бути відкритою системою і мати властивість зручності включення нових методів і засобів.
5. САПР повинна бути об'єктно-орієнтованою системою з максимальним використанням уніфікованих модулів. Необхідна умова уніфікації – пошук загальних ознак і положень при аналізі, синтезі і тестуванні різнорідних технічних об'єктів.

Основні концепції реалізуються завдяки використанню визначених принципів побудови САПР, до яких відносяться:

- сумісність ручного, автоматизованого й автоматичного режимів проектування;
- автономність різних частин системи і ієрархічність її програмних засобів, що дозволяє здійснювати просте розширення останніх при вдосконаленні САПР;
- інтерактивність системи;
- найменший час, необхідний для взаємодії системи з користувачем, що забезпечується наявністю оперативного технологічного обладнання і пристроїв введення-виведення, досконалістю засобів взаємодії із системою, форм подання результатів проектування, їхньою інформативністю і т. д.;
- простота експлуатації САПР, що досягається завдяки можливості одержання в режимі діалогу різноманітної довідкової інформації, яка використовується для підказки альтернативних рішень, прийнятих у ході автоматизованого проектування;
- наявність засобів контролю вхідної і вихідної інформації;
- інваріантність структури системи до змін параметрів базових елементів;
- переналагоджуваність системи (можливість керування окремими етапами проектування);
- наявність засобів автоматичного внесення змін у проект;
- можливість видачі необхідної інформації за запитом проектувальника;
- можливість вибору різних за інформативністю мовних засобів описів вихідної інформації, гнучкість їхнього використання.

У структурному плані САПР включають підсистеми, виділені за деякими принципами складові частини, що забезпечують одержання

закінчених проектних рішень і відповідних проектних документів. Розрізняють підсистеми, що *проектують* і *обслуговують*, які, у свою чергу, поділяють на об'єктно-орієнтовані й об'єктно-незалежні (інваріантні). Підсистеми, що проектують, призначені для рішення задач окремих етапів або рівнів, а ті, що обслуговують – виконують допоміжні функції.

У посібнику докладно викладені загальні питання організації процесів розробки засобів обчислювальної техніки і можливості їхньої автоматизації. Особливу увагу приділено опису основного компонента САПР – програмному забезпеченню.

1 ЕОМ ЯК ОБ'ЄКТ ПРОЕКТУВАННЯ

1.1 Етапи та рівні проектування ЕОМ, ступінь і можливості їх автоматизації

При побудові нових об'єктів техніки по заданому опису не існуючого об'єкта виконується його матеріалізація в працездатну надійну конструкцію.

Проектування – це процес створення опису, необхідного для побудови в заданих умовах ще не існуючого об'єкта, на основі первинного опису цього об'єкта.

Процес створення опису нового об'єкта може виконуватися різними способами. Виділимо три основних.

1. Якщо весь процес проектування здійснює людина, то проектування називають *неавтоматизованим*. В наш час неавтоматизоване проектування таких складних об'єктів, як електронно-обчислювальна апаратура (ЕОА), практично не використовують.

2. Найбільшого поширення набуло проектування, при якому відбувається взаємодія людини й ЕОМ. Таке проектування називають *автоматизованим*. Автоматизоване проектування, як правило, здійснюється в режимі діалогу людини з ЕОМ на основі застосування спеціальних мов спілкування з ЕОМ.

3. Проектування, при якому всі перетворення описів об'єкта й алгоритму його функціонування здійснюються без участі людини, називають *автоматичним*.

Розглянемо ряд понять, що використовуються, наприклад, при розробці ЕОА. Первинний опис ЕОА, поданий в заданій формі, називається *завданням на проектування*. У завданні на проектування ЕОА повинна бути інформація про призначення ЕОА, її параметри, способи функціонування, конструктивну реалізацію, умови виготовлення і т.п.

Проектним рішенням називається проміжний чи кінцевий опис об'єкта проектування, необхідний і достатній для розгляду і визначення подальшого напрямку проектування чи його закінчення.

Проектне рішення, чи їхня сукупність, що задовольняє заданим вимогам, необхідне для створення об'єкта проектування, буде результатом проектування. В задані вимоги повинні бути обов'язково включені вимоги до форми проектного рішення, що подається.

Документ, виконаний за заданою формою, в якому подане будь–яке проектне рішення, отримане при проектуванні, називається *проектним*. Сукупність проектних документів, відповідно до встановленого переліку, у яких поданий результат проектування, називається *проектом*.

Під ***проектною процедурою*** розуміють формалізовану сукупність дій, виконання яких закінчується проектним рішенням. Наприклад, проектними процедурами є оптимізація, контроль, пошук рішення, корегування, компонування і т.п.

Дія, чи формалізована сукупність дій, які складають частину проектної процедури, алгоритм виконання яких залишається незмінним для ряду проектних процедур, називається ***проектною операцією***.

Прикладами проектних операцій є складання топології, введення і виведення даних і т.п. Відповідно, проектна процедура, алгоритм якої залишається незмінним для різних об'єктів, чи реалізація різних стадій проектування того ж самого об'єкта, називається *уніфікованою* проектною процедурою.

При створенні нових об'єктів виділяють такі етапи:

1. *Етап науково-дослідних робіт (НДР).*

Цей етап поєднує такі стадії:

- передпроектні дослідження;
- розробка технічного завдання;
- частина технічних пропозицій.

На цьому етапі проводяться дослідження з пошуку нових принципів функціонування, нових структур, фізичних процесів, нової елементної бази, технічних засобів і т.п.

2. Етап дослідно-конструкторських робіт (ДКР).

Цей етап поєднує такі стадії:

- частина технічних пропозицій;
- ескізний проект;
- технічний проект.

На цьому етапі відпрацьовуються питання детального конструкторського опрацювання проекту.

1. Етап робочого проектування.

Поєднує стадії:

- робочий проект;
- виготовлення;
- налагодження;
- випробування;
- впровадження в дію.

На цьому етапі здійснюються схемні, конструкторські і технологічні рішення, проводять випробовування та виготовлення фізичного зразка.

Для реалізації етапу НДР, в основному, використовують системи автоматизації наукових досліджень і експериментів.

Розподіл робіт між підрозділами проводять з використанням *блочно-ієрархічного підходу* (БІП) до проектування. Цей підхід заснований на

структуруванні описів об'єкта з поділом опису на ряд ієрархічних рівнів за ступенем детальності відображення в них властивостей об'єкта і його частин. Кожному ієрархічному рівню характерні свої форми документації, математичний апарат для побудови моделей і алгоритмів дослідження.

Сукупність мов, моделей, постановок задач, методів одержання описів деякого ієрархічного рівня часто називають ***рівнем проектування.***

Рівні проектування можна виділяти не тільки за ступенем деталізації відображення властивостей об'єкта, але і за характером відбиваних властивостей. Якщо в першому випадку рівні називають *горизонтальними* чи ієрархічними, то в другому — *вертикальними* чи аспектами.

Методологія БІП базується на трьох концепціях:

- розбивка і локальна оптимізація;
- абстрагування;
- повторюваність.

Концепція *розбивки* дозволяє складну задачу проектування об'єкта звести до розв'язання більш простих задач з урахуванням взаємодії між ними. *Локальна оптимізація* має на увазі поліпшення параметрів усередині кожної простої задачі. *Абстрагування* полягає в побудові формальних математичних моделей, що відбивають тільки значимі, в даних умовах, властивості об'єктів. *Повторюваність* полягає у використанні існуючого досвіду проектування.

Основна перевага БІП – це спрощення процесу проектування й одержання можливості розв'язувати задачі проектування доступними засобами.

Використання БІП допомагає:

- спростити вирішення проблеми збереження даних;
- скоротити розмірність виконуваних програм і час проектування;
- застосовувати САПР для проектування об'єкта (його частини) незалежно від числа ідентичних об'єктів (його частин).

Весь процес проектування можна подати як послідовність етапів, що зв'язують концептуальний опис об'єкта і створення цього об'єкта. Зазначений зв'язок реалізують в одному з двох напрямків: *висхідному* чи *спадному*. Висхідне проектування, тобто проектування знизу вверх, характеризується рішенням спочатку задач низьких ієрархічних рівнів з послідовним переходом до рішення задач вищих рівнів. Спадне проектування, тобто проектування зверху вниз, є протилежним стосовно висхідного проектування.

Відзначимо, наприклад, що використовувана на сьогодні концепція проектування інтегральних мікросхем з великим ступенем інтеграції за модульним принципом – це концепція БІП. У системі БІП конструктор виконує функціональні, інтуїтивні й інтелектуальні перетворення на

верхніх рівнях, а ЕОМ виконує проектування на нижніх рівнях.

У загальному випадку, при проектуванні технічних об'єктів, можна виділити кілька вертикальних рівнів, основні з них – функціональний, конструкторський, технологічний. Опис кожного вертикального рівня, у свою чергу, теж поділяють на ієрархічні рівні. В таблиці 1.1 наведено приклад структурування опису ЕОМ.

Функціональне проектування містить у собі аналіз технічного завдання (ТЗ) і на його основі вибір із системних позицій методики побудови і шляхів реалізації обчислювального процесу в ЕОА. Воно зв'язано з аналізом і синтезом блоків ЕОА і полягає в розробці функціональних і принципових схем При цьому визначають принципи функціонування і найважливіші параметри і характеристики ЕОА.

Таблиця 1.1 – Структурування опису ЕОМ

<table>
<tr><td rowspan="2">Горизонтальні рівні</td><td colspan="4">Вертикальні рівні</td></tr>
<tr><td>Функціональний</td><td>Алгоритмічний</td><td>Конструкторський</td><td>Технологічний</td></tr>
<tr><td>Системний</td><td rowspan="2">Програмування системи</td><td>Шафа, стійка</td><td rowspan="3">Принципова схема технологічного процесу</td></tr>
<tr><td>Логічний</td><td rowspan="2">Панель</td></tr>
<tr><td rowspan="2">Схемотехнічний</td><td rowspan="2">Програмування модулів</td></tr>
<tr><td>Типовий елемент заміни</td><td>Маршрутна технологія</td></tr>
<tr><td rowspan="3">Компонентний</td><td rowspan="3">Проектування мікропрограм</td><td>Модуль</td><td rowspan="3">Технологічні операції</td></tr>
<tr><td>Кристал</td></tr>
<tr><td>Комірка</td></tr>
</table>

Основні задачі функціонального проектування такі:

1. Розробка структурних схем, визначення вимог до вихідних параметрів.

2. Аналіз і формування ТЗ на розробку окремих блоків ЕОА з подальшим синтезом функціональних і принципових схем отриманих блоків.

3. Контроль і виробництво діагностичних тестів і перевірка працездатності синтезованих блоків.

4. Розрахунки параметрів пасивних компонентів і визначення вимог до параметрів активних компонентів.

5. Формулювання ТЗ на проектування компонентів.

6. Вибір фізичної структури, топології компонентів.

7. Розрахунки параметрів дифузійних профілів і напівпровідникових компонентів, електричних параметрів, параметрів технологічних процесів, дифузії, окислювання і т.д.

8. Імовірнісні вимоги до вихідних параметрів компонентів.

Алгоритмічне проектування полягає в розробці алгоритмів функціонування та створення математичного забезпечення ЕОА.

Конструкторське проектування полягає в реалізації принципових схем у заданому конструктивному базисі. При цьому зважуються питання вибору форм і матеріалів, вибору типорозмірів, компонування, розміщення елементів, трасування з'єднань, контролю.

Основні задачі конструкторського проектування такі:

1. Покриття функціональних схем, тобто одержання принципових електричних схем.
2. Конструкторський розрахунок геометричних розмірів компонентів і площі розміщення.
3. Компонування елементів.
4. Розміщення елементів з розрахунком конструкторських схемотехнічних і технологічних обмежень.
5. Трасування з'єднань.
6. Контроль топології.
7. Проектування фотошаблонів.
8. Випуск конструкторсько-технологічної документації.

Технологічне проектування полягає в рішенні задач технологічної підготовки виробництва, розробці принципових схем, маршрутів, операцій і переходів технологічних процесів виготовлення деталей, зборки і монтажу вузлів, включаючи вибір оснащення, інструмента, технологічного обладнання і т.п.

Функціональне проектування ЕОА складається з чотирьох основних горизонтальних рівнів:

- системного;
- логічного;
- схемотехнічного;
- компонентного.

На *системному* рівні визначають загальну структурну схему, структурні схеми основних блоків.

На *логічному* рівні створюють функціональні і принципові схеми ЕОА. Тут виділяють підрівні – *регістровий* і *вентильний*. На регістровому підрівні проектуються пристрої з модулів (функціональних вузлів) типу регістрів, лічильників, суматорів, інтеграторів тощо. На вентильному рівні проектуються пристрої і модулі з окремих логічних вентилів і тригерів.

Алгоритмічне проектування використовується для розробки програмного забезпечення ЕОА. Для великих програмних систем звичайно використовують набір ієрархічних рівнів, два з який є основними. На першому планують усю програмну систему і розробляють схеми алгоритмів на основі програмних модулів. На другому роблять програмування модулів заданою алгоритмічною мовою.

Конструкторське проектування складається з ієрархічних рівнів проектування компонентів, великих інтегральних схем (ВІС), типових елементів заміни (ТЕЗ), панелей, стійок, шаф. При цьому, в основному, використовується висхідне проектування.

Технологічне проектування складається з рівнів проектування принципової схеми технологічного процесу, технологічних маршрутів, технологічних операцій.

1.2 Методи проектування: задача синтезу та аналізу

Аналіз технічних об'єктів у САПР заснований на математичному моделюванні, тобто на дослідженні проектованих об'єктів шляхом оперування їхніми математичними моделями.

Таким чином, математичні моделі об'єктів проектування на мікро- і макрорівнях зводяться до систем звичайних диференціальних і кінцевих рівнянь (під кінцевими рівняннями розуміються алгебраїчні і трансцендентні рівняння). Оперування такими моделями в процедурах одноваріантного аналізу означає розв'язання відповідних рівнянь. Тому методи одноваріантного аналізу на цих рівнях – суть чисельні методи розв'язання систем диференціальних і кінцевих рівнянь, теж відносяться до моделей і методів аналізу аналогової радіоелектронної апаратури (РЕА) на метарівні.

Різноманітний аналіз полягає в багаторазовому повторенні розв'язання систем названих рівнянь при варіюванні внутрішніми і (чи) зовнішніми параметрами. Типовими процедурами різноманітного аналізу, які реалізовано в САПР, є процедури аналізу чутливості і статистичного аналізу.

Особливістю цифрової РЕА є використання на функціонально-логічному рівні проектування як моделі – систем логічних рівнянь, а як методів аналізу – методів розв'язання цих систем.

В обчислювальній математиці відома велика кількість методів чисельного розв'язання систем рівнянь, але застосування більшості з них у САПР РЕА є неефективним, що пов'язано з особливостями математичних моделей об'єктів, які проектуються. Тому при створенні математичного забезпечення САПР зусилля спрямовані не тільки на розробку математичних моделей, але й на розвиток чисельних методів і алгоритмів аналізу. Оскільки ефективність методу залежить від особливостей розв'язуваної задачі, доцільна реалізація більш ніж одного методу для кожного класу рівнянь. Вибір методу, в більшості випадків, покладається на користувача, який повинен мати відповідну підготовку в галузі чисельних методів аналізу.

При невдалому виборі моделей чи методів аналізу користувач САПР може зіштовхнутися з цілою низкою неприємностей: надмірною

тривалістю обчислень, нестійкістю обчислювального процесу, малою точністю одержаних результатів і т.д.

У САПР доцільно застосовувати методи, що виключають можливість виникнення подібних ситуацій, тобто методи, які володіють властивостями високої економічності, надійності і точності. Однак ці вимоги суперечливі і не завжди вдається задовольняти їм належною мірою, тому важливо вміти розпізнавати несприятливі ситуації і знати фактори, зміна яких може привести до виправлення положення.

Економічність методу характеризується витратами обчислювальних ресурсів (машинного часу $T_м$ і машинної пам'яті $П_м$) на його застосування в деяких, заздалегідь визначених умовах (наприклад, у тестових задачах, у середньому по групі задач визначеного класу і т.п.). На показники $T_м$ і $П_м$ звичайно впливають багато факторів і, в першу чергу, розмірність N розв'язуваної задачі. За N приймають порядок розв'язуваної системи рівнянь, число елементів, з яких складається об'єкт моделювання і т.п.

При порівнянні методів за економічністю часто не цікавляться абсолютними показниками $T_м$ і $П_м$ у конкретній ситуації, а досліджують характер залежності від N. Найефективніші методи мають лінійну, чи близьку до лінійної, залежність показників економічності від складності задачі. Для багатьох чисельних методів характерна поліноміальна залежність $T_м$, яка накладає помітне обмеження на складність розв'язуваних задач.

Надійність методу оцінюється як ймовірність одержання правильних результатів при використанні методу для розв'язання задач заданого класу. Звичайно умови застосовності методу пов'язані з такими характеристиками математичних моделей аналізованих об'єктів, які користувач не може оцінити заздалегідь наявними в його розпорядженні засобами, тому можливі ситуації, коли обчислювальний процес виявляється хистким чи відсутня його збіжність, що може виражатися в зациклюванні чи зупинці ЕОМ через переповнення розрядної сітки. У САПР намагаються застосовувати надійні методи. Однак високонадійні методи часто характеризуються недостатньою економічністю. У цьому випадку доцільне комбінування методів з переходом до трудомістких, але надійних методів тільки в результаті автоматичного розпізнавання ситуацій незбіжності чи нестійкості обчислень.

Точність розв'язання задачі визначається особливостями використаних моделей, чисельних методів, обмеженістю розрядної сітки ЕОМ. Кожне джерело похибки повинно контролюватися для того, щоб похибки не перевищили гранично припустимих значень. Звичайно точність результатів, одержаних за допомогою чисельного методу, залежить від деяких параметрів, обраних «за замовчуванням» чи задається серед вихідних умов. За допомогою цих параметрів можна керувати похибками рішення, але необхідно пам'ятати, що зниження похибок

можливе лише до деякої, відмінної від нуля, межі і, як правило, супроводжується збільшенням витрат машинного часу. Доцільно в математичному забезпеченні САПР мати не один, а декілька методів однакового цільового призначення, але з різними можливостями компромісного задоволення суперечливих вимог точності й економічності.

Користувач САПР повинний також знати, що явища зациклювання обчислень чи переповнення розрядної сітки можуть відбуватися не тільки через недоліки обраного чисельного методу, але і через помилки в заданні вихідних даних. Деякі помилки, пов'язані з порушенням формальних правил граматики вхідної мови, розпізнаються автоматично. Однак ряд помилок не може бути виявлений формальними засобами без участі користувача. Прикладами таких помилок можуть бути помилки в заданні чисельних значень чи параметрів, у заданні з'єднань в схемі, що аналізується. Якщо ці помилки призводять до одержання моделі самозбудної схеми, то можливі явища зациклювання і переповнення розрядної сітки.

Великорозмірні задачі проектування, необхідність виконання багатьох варіантів розв'язання систем рівнянь при проектуванні ЕОМ і інших складних технічних об'єктів обумовлюють великі витрати обчислювальних ресурсів. Тому підвищення економічності методів аналізу при дотриманні вимог точності є актуальною задачею створення й удосконалення математичного забезпечення САПР. Ця задача базується на основі ідей і методів, які згруповані у декілька напрямків.

Декомпозиція – розподіл моделі проектованого об'єкта на частини і нарізний аналіз цих частин. Однак нарізний аналіз відбувається в умовах прийняття припущень, що спрощують взаємний вплив частин, тобто супроводжується збільшенням похибок розрахунків. Декомпозиція є основою блочно-ієрархічного підходу до проектування. Цей напрямок широко використовують як в автоматизованих, так і в неавтоматизованих методах проектування.

Діакоптика – напрямок дослідження складних систем частинами, що відрізняється від декомпозиції тим, що роздільний аналіз здійснюється без припущень, які спрощують вплив частин одна на одну. Економічність діакоптичних методів порівняння з економічністю звичайних декомпозиційних методів, а точність – вища.

Облік розрідженості матриць – напрямок економічної організації операцій над розрідженими матрицями. Матрицю називають розрідженою, якщо в ній переважають нульові елементи. Відмова від збереження нульових елементів і реалізація алгоритмів, у яких ігноруються арифметичні дії над нульовими елементами, можуть дати значну економію $T_м$ і $П_м$.

Врахування неактивності – напрямок, називаний також врахуванням тимчасової розрідженості моделей, який засновано на виключенні з

обчислювального процесу дій над неактивними змінними. Неактивною на деякому інтервалі змінною називають величину, зміни якої на цьому інтервалі не перевищують досить малого раніше заданого значення. У моделях складних систем у кожен момент модельованого часу більшість змінних неактивна. Моделювання, засноване на врахуванні неактивності, прийнято називати неактивним моделюванням. В алгоритмах неактивного моделювання необхідно реалізувати критерії своєчасного включення змінних і відповідних їм частин моделей у групу неактивних (латентних) і своєчасного їхнього виключення з цієї групи.

Комбінування моделей і методів – одночасне використання при розв'язанні конкретної задачі декількох різнотипних моделей чи методів аналізу однакового цільового призначення. Комбінування може бути просторовим, якщо різнотипні моделі чи методи застосовують у різних частинах загальної моделі, чи часовим, якщо їх застосовують на різних етапах обчислювального процесу. Просторове комбінування є окремим випадком діакоптичного підходу, тому що здійснюється поділ моделі на частини (фрагменти). Підвищення ефективності при комбінуванні моделей і методів базується на використанні найпридатніших моделей і методів для даного фрагмента і даного етапу обчислень. Просторове комбінування моделей, що відносяться до різних ієрархічних рівнів, називають багаторівневим (чи змішаним) моделюванням.

Поняття «***синтез***» технічного об'єкта в широкому сенсі слова близько за змістом до поняття «проектування». Задача синтезу технічного об'єкта полягає в тому, щоб за заданим функціональним призначенням об'єкта, чи за законом його функціонування, одержати проектне рішення у вигляді деякого опису проектованого об'єкта.

Синтез технічних об'єктів має за мету створення нових варіантів, а аналіз використовується для оцінювання цих варіантів, тобто синтез і аналіз виступають у процесі проектування в діалектичній єдності.

При синтезі заздалегідь заданий припустимий набір використовуваних елементів (електро-радіоелементів при синтезі електронних схем, набір балок і блокових конструкцій при проектуванні будівельних споруд і т.д.), можливі правила їхнього поєднання між собою і способи визначення за синтезованою структурою об'єкта функції, що він реалізує.

Під *структурою* об'єкта розуміється набір його елементів і зв'язків між ними. Структура визначає, як влаштований об'єкт проектування (ЕОМ, автоматична система, радіотехнічний комплекс і т.д.), з яких фізичних частин він складається і як ці частини пов'язані між собою.

До об'єктів проектування відносяться структура і конструкція, причому під конструкцією об'єкта розуміється матеріалізована сукупність з'єднаних між собою елементів, що виконують задані функції. Зокрема,

конструкція ЕОМ – це сукупність електричних і механічно з'єднаних елементів, у якій реалізується електрична схема даної машини.

На структуру і конструкцію будь-якого проектованого об'єкта завжди накладається безліч різних обмежень. При цьому одна група обмежень відноситься до методу розв'язання задачі й охоплює такі питання, як наявність знань, терміни і наявні в розпорядженні технічні засоби проектування. Інша група обмежень пов'язана з вимогами технічного завдання на параметри проектованого об'єкта, з вимогами стандартів і технології виготовлення вузлів і різних елементів об'єкта. Третя група обмежень формується фізичними принципами реалізації закону функціонування об'єкта й одержання його бажаних характеристик. Додаткові обмеження накладаються способами і формами взаємодії проектованого об'єкта з зовнішнім середовищем, а також методами організації взаємодії людини з проектованим об'єктом у процесі функціонування й експлуатації.

При проектуванні будь-якого технічного об'єкта обираються чи задаються в технічному завданні на проектування критерії оптимальності. Залежно від характеру і призначення проектованого об'єкта як критерій оптимальності може бути прийнята його вартість, коефіцієнт корисної дії, споживана потужність чи інший, більш складний критерій.

Аналіз технічних об'єктів передбачає вивчення їх властивостей.

При проектуванні технічних об'єктів важливе значення має визначення оптимальних варіантів структур і конструкцій машин і пристроїв, параметрів схем, режимів роботи технологічного обладнання і т.д. Під оптимальним будемо розуміти такий варіант структури чи конструкції, параметри якої задовольняють усім системним, конструктивним, технологічним, електричним і економічним вимогам технічного завдання, а критерій оптимальності, що описує якість проектованої структури чи конструкції, набуває найкращого (мінімального чи максимального) значення.

Одержання оптимальних рішень при проектуванні стало можливим і досяжним як за термінами проектування, так і за вартістю реалізації проектних процедур тільки при автоматизованому проектуванні, коли з'являється можливість синтезувати і досліджувати множину варіантів структур і конструкцій, а також проводити кількісне вивчення проектованих технічних об'єктів, що у минулому вивчалися лише якісно.

Прикладами задач оптимального проектування є визначення структури ЕОМ максимальної продуктивності при заданих вагових і габаритних обмеженнях, надійності, споживаній потужності; розрахунок елементів конструкцій літального апарата максимальної вантажопідйомності при заданій потужності двигуна й обмеженнях на інші параметри апарата; визначення конструктивних параметрів електричних двигунів, оптимальних за критерієм мінімальної вартості і ін.

Процес проектування (рис. 1.1.) можна звести до взаємодії процедур синтезу і аналізу.

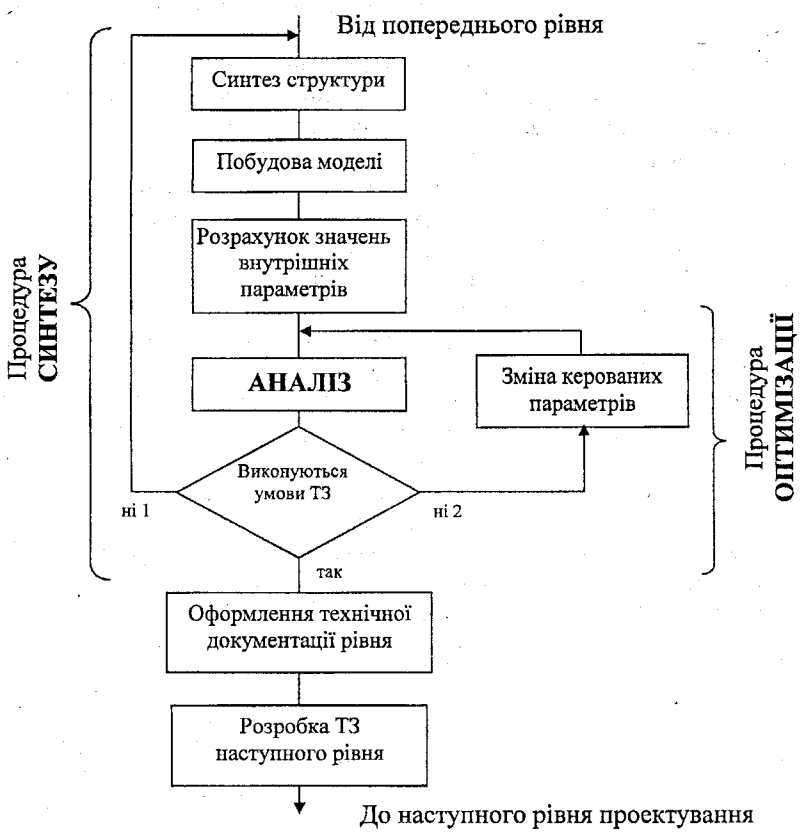


Рисунок 1.1 – Взаємодія процедур синтезу та аналізу

Будь-якому варіанту проектованого об'єкта відповідає своя структура і конструкція. При автоматизованому проектуванні для породження безлічі альтернативних структур технічного об'єкта, еквівалентних за функціональним призначенням, але різних за тактико-технічними характеристиками, необхідна розробка математичної моделі об'єкта, яка є формальним описом проектованого об'єкта на прийнятому рівні деталізації.

Для розв'язання задачі синтезу технічних об'єктів виділяють деяку сукупність незалежних змінних $X=(x_1 \ldots, x_m)$, фіксація значень яких визначає один з варіантів об'єкта і його кількісні характеристики, у тому

числі значення критерію оптимальності, а також показників, прийнятих як обмеження.

Змінні $x_1, ..., x_m$ називають змінними проектування й залежними від фізичної природи об'єкта проектування. Вони мають різну інтерпретацію, зокрема, можуть характеризувати кількість вузлів кожного типу в об'єкті, вказувати на включення чи на не включення того чи іншого вузла в структуру об'єкта, нести інформацію про геометричні розміри виробу і т.д.

У багатьох задачах математичного програмування деякі змінні можуть набувати лише визначених дискретних значень (наприклад, діаметр обмотувального проводу, вибраний з певного стандарту, номінали конденсаторів і т.д.) або тільки цілочисельних значень (наприклад, число верстатів, що випускаються, літаків і т.д.). У цьому випадку задача проектування може бути сформульована в термінах дискретного програмування.

Інша особливість задачі математичного програмування полягає в тому, що в загальному випадку нелінійна цільова функція $F(x)$ може мати декілька локальних екстремумів у припустимій області, включаючи її границю. Тому при оптимальному проектуванні структур і конструкцій технічних об'єктів виникає проблема визначення глобального екстремуму задачі оптимізації, що в практичних задачах проектування нерідко викликає значні труднощі при обчисленнях.

У загальному вигляді задача нелінійного програмування поки не має строгого математичного розв'язання. Однак, у зв'язку з тим, що даний клас задач досить часто зустрічається в практичних задачах проектування, розроблено велику кількість методів і евристичних алгоритмів розв'язання конкретних задач нелінійного програмування.

Контрольні запитання

1. Мета та задачі автоматизації проектування.
2. Передумови та можливості автоматизації проектування ЕОМ.
3. Роль ЕОМ в процесі розв'язання інженерних задач проектування.
4. Характеристика етапів проектування.
5. Етапи та рівні проектування ЕОМ, ступінь і можливості їх автоматизації.
6. Основні задачі проектування.
7. Взаємодія синтезу та аналізу при проектуванні обчислювальних засобів.

2 ЗАГАЛЬНІ УЯВЛЕННЯ ПРО САПР

2.1 Поняття системи автоматизованого проектування

Розглянемо загальні питання організації автоматизованого проектування. Кожна САПР має засоби описування об'єктів і задач проектування. Отримані результати повинні бути певним чином інтерпретовані (визначені).

Схему реалізації автоматизації проектування наведено на рисунку 2.1.

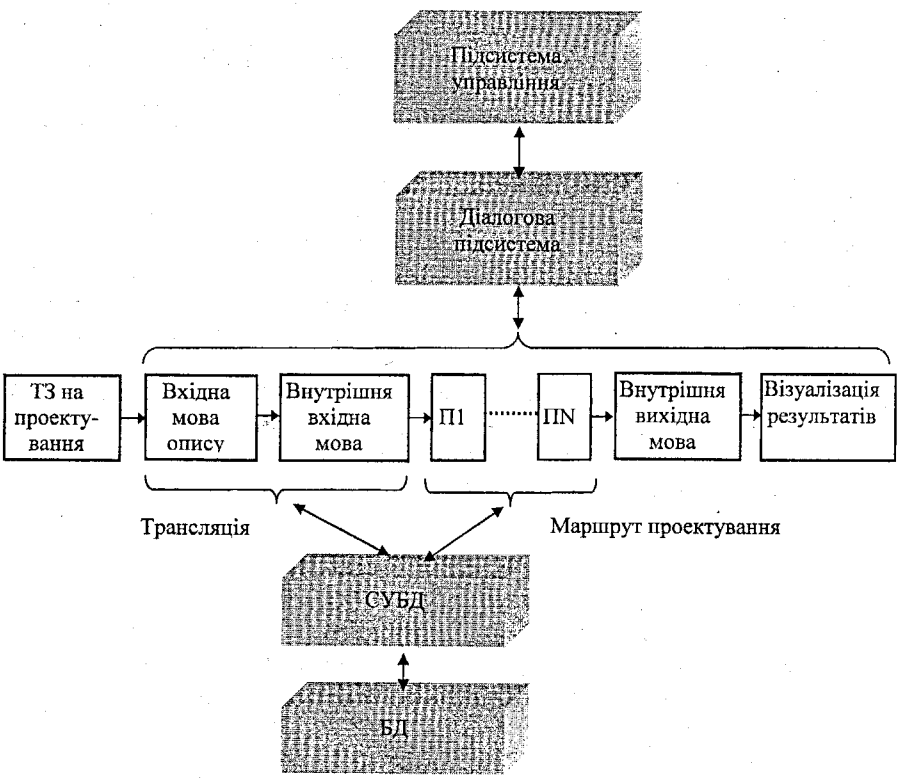


Рисунок 2.1 – Схема реалізації автоматизованого проектування

Спочатку формулюється завдання на проектування у вигляді ТЗ, яке повинне враховувати можливості його подання вхідною мовою системи. Так, при проектуванні пристроїв логічного керування (ПЛК) завданням може бути граф-схема алгоритму (ГСА), який повинен реалізувати побудований ПЛК. Потім здійснюється подання вихідних даних вхідною мовою, передбаченими засобами лінгвістичного забезпечення САПР. Ця

мова схожа на широко розповсюджені алгоритмічні мови і дає можливість виразити у формі, зрозумілій для ЕОМ, усі ті вимоги, які проектувальник висуває до нового об'єкта. Вдосконалювання пристроїв підготовки інформації для ЕОМ дозволяє вводити вихідну інформацію в звичному графічному вигляді. З цією метою використовуються спеціальні графічні мови і засоби підготовки даних, які називаються *графічними редакторами.* Останні, наприклад, реалізовані в прикладній системі PCAD, Active-HDL для ПЕОМ сім'ї IBM PC. Як засоби підготовки графічних даних використовуються термінали, що включають розширену клавіатуру і графічний дисплей з високою роздільювальною здатністю, пристрої визначення координат, маніпулятори типу "мишка" для ПЕОМ і т п.

Вихідні дані подані вхідною мовою в такій формі, що легко сприймається людиною. Для обробки в машині вони, звичайно, незручні. Тому ці дані спочатку перекладаються у внутрішню форму подання, що вибирається з уявлень підвищення ефективності обробки інформації засобами ПЕОМ. Процедура такого перекладу називається *трансляцією.* У процесі її виконання можуть бути виявлені синтаксичні і семантичні помилки вихідного опису. Вхідна інформація внутрішньою вхідною мовою подає вихідні дані для прикладних програм. Послідовне виконання цих програм приводить до реалізації заданого *маршруту проектування.* Отримані результати спочатку подаються внутрішньою вихідною мовою, а потім виводяться на відповідні пристрої візуалізації у формі, зрозумілій проектувальнику.

Наприклад, при проектуванні ПЛК може бути виведена відповідна принципова електрична схема. Як периферійне обладнання тут використовуються графічні й алфавітно-цифрові дисплеї, графопобудовники, принтери і т. д. Останнім часом починають широко застосовуватися сполучені з обчислювальним обладнанням технологічні пристрої і робототехнічні комплекси, що дозволяють автоматично виготовляти відповідні вироби.

Для організації введення-виведення інформації, взаємодії прикладних програм у заданому маршруті проектування використовується *підсистема управління* (операційна система САПР чи керувальний монітор). В її функції входять:

- Інтерпретація завдань користувача.
- Визначення послідовності виконання програм.
- Встановлення способів взаємодії програм з базою даних.
- Побудова структур з перекриттями (оверлейних програм).
- Виконання сервісних процедур.

Підсистема управління взаємодіє з *діалоговою підсистемою,* яка забезпечує необхідний інтерфейс із користувачем. Вона певним чином сприймає вказівки проектувальника і знаходить для нього необхідну інформацію або виконує задані дії. Діалогова система дозволяє швидко

усунути можливі синтаксичні і семантичні помилки, що виникають на етапі трансляції, переглянути і, при необхідності, скорегувати результати роботи окремих програм і т. п. З її допомогою можна одержати довідкову інформацію, наприклад, про параметри використовуваних базових елементів або про правила підготовки вихідних даних вхідною мовою.

При виконанні автоматизованого проектування велике значення мають різні дані. Фактично будь-яка прикладна програма здійснює перетворення даних. Вона одержує їх від іншої програми або з попереднього пристрою. Програми, які взаємодіють між собою, повинні бути інформаційно-узгодженими. У найпростішому випадку вони можуть бути жорстко пов'язані. При цьому підсистема управління директивно вказує на те, що після програми n повинна бути виконана деяка інша програма $n + 1$.

Вихідні дані першої з них є вхідними для другої. Зміна маршруту проектування призводить до необхідності корегування керувальною системою послідовності викликаних модулів і операторів оголошення даних. При іншому підході результати роботи однієї програми заносяться до бази даних, а інша програма забирає і використовує їх. Підсистема управління визначає тільки порядок підключення програм. Це дозволяє організувати в режимі діалогу конструювання нових маршрутів проектування з наявних програмних модулів. Тут можна розглядати конкуруючі набори модулів, здійснювати просте розширення і модифікацію програмних засобів САПР і т. п.

Дані, зазвичай, зберігаються в базі на магнітних носіях інформації. Взаємодія з прикладними програмами здійснюється через систему управління базою даних (СУБД). Дані, що зберігаються, певним чином класифікуються залежно від вимог, висунутих до системи проектування. Наприклад, виділяються довідкова інформація, архівна інформація, призначена для багатьох користувачів, і т. п.

Обробка даних у кожній прикладній програмі виконується відповідно до заданого алгоритму, що реалізує методи і засоби, включені до складу математичного забезпечення САПР.

Усі визначення, наведені нижче, дані відповідно до існуючого ДСТ і стандартів з автоматизації проектування.

Система автоматизованого проектування – організаційно-технічна система, що складається з комплексу засобів автоматизації проектування (КЗАП), пов'язаного з необхідними підрозділами проектної організації чи колективом фахівців (користувачів системи) і виконує автоматизоване проектування. Відповідно ***система автоматичного*** проектування виконує автоматичне проектування без участі людини.

2.2 Види забезпечення САПР

Комплекс засобів автоматизації проектування – це сукупність різних видів забезпечення автоматизованого (автоматичного) проектування, необхідних для виконання АП.

Математичне забезпечення (МЗ) САПР – це сукупність математичних методів (ММет), математичних моделей (ММ) і алгоритмів проектування (АлП), необхідних для виконання АП, поданих у заданій формі.

Технічне забезпечення (ТЗаб) САПР – це сукупність пов'язаних і взаємодіючих технічних засобів, призначених для виконання АП.

Технічні засоби виконують визначену функцію в САПР і є певними компонентами ТЗаб. До ТЗаб відносяться пристрої обчислювальної й організаційної техніки, засоби передачі даних, вимірювальні й інші пристрої. Розрізняють такі групи технічних засобів:

- *Підготовка і введення даних.* Група призначена для автоматизації підготовки і редагування даних при введенні в ЕОМ алфавітно-цифрової і графічної інформації. Група надає можливість кодування інформації, перенесення даних на машинні носії, введення даних в ЕОМ, візуального контролю і редагування даних при введенні інформації.
- *Передача даних.* Група призначена для забезпечення дистанційного зв'язку технічних засобів телефонними, телеграфними і спеціальними каналами зв'язку.
- *Програмна обробка даних.* Група містить у собі універсальні чи спеціалізовані ЕОМ, що забезпечують прийом цифрових даних із пристроїв введення чи каналів зв'язку, їх програмної обробки, накопичування і виведення на машинні носії.
- *Пристрої відображення і канали зв'язку.* Дана група дозволяє змінити продуктивність шляхом нарощування ЕОМ, використовувати мультипрограмний і діалоговий режими роботи.
- *Відображення і документування даних.* Група призначена для оперативного подання і документування проектних рішень. Тут використовують принтери і графопобудовники, мікрофільми і пристрої відображення візуальної інформації.
- *Архів проектних рішень.* Група призначена для забезпечення збереження, контролю, відновлення і розмноження даних про проектні рішення і довідкові дані.

Компоненти ТЗаб створюються на базі серійних засобів обчислювальної техніки загального призначення і спеціалізованих технічних засобів. В наш час, переважно, використовують дворівневу ієрархічну структуру комплексу ТЗаб САПР. Структура містить у собі компоненти *центрального обчислювального комплексу* (ЦОК) і компоненти *термінального комплексу* (ТК). ЦОК будують на базі ЕОМ,

обчислювальних систем і мереж ЕОМ колективного користування. Термінальний комплекс САПР будують на основі *автоматизованих робочих місць* (АРМ), *термінальних станцій* з використанням мікропроцесорів, міні- і мікро-ЕОМ.

Програмне забезпечення (ПЗ) САПР – сукупність машинних програм, необхідних для виконання АП, поданих у заданій формі. Частину ПЗ САПР, призначену для керування проектуванням, називають *операційною системою* (ОС) САПР.

Сукупність машинних програм, необхідних для виконання будь-якої проектної процедури і поданих у заданій формі, називають *пакетом прикладних програм* (ППП).

Компонентами ПЗ є документи з текстами програм, програми на всіх видах носіїв, експлуатаційні документи. Програмне забезпечення поділяють на загальносистемне (ЗПЗ) і прикладне (ППЗ). Компонентами ЗПЗ є транслятори з алгоритмічних мов, емулятори, супервізори і ін. Компонентами ППЗ є програми і пакети прикладних програм для АП.

Інформаційне забезпечення (ІЗ) САПР – сукупність відомостей, необхідних для виконання АП, поданих у заданій формі. Основною частиною ІЗ САПР є *автоматизовані банки даних,* що складаються з баз даних (БД) САПР і систем управління базами даних (СУБД). До ІЗ САПР входять нормативно-довідкові документи, завдання державних планів, прогнози технічного розвитку, типові проектні рішення, системи класифікації і кодування техніко-економічної інформації, системи документації типу ЄСКД, ЄСТД, файли і блоки даних на машинних носіях, фонди (нормативні, планові, прогнозні) типових рішень, алгоритмів і програм і т.п.

Керування автоматизованим банком даних здійснюють проектувальники, при цьому необхідно забезпечити цілісність, правильність даних, ефективність і функціональні можливості СУБД. Проектувальник організовує і формує БД, визначає питання її використання і реорганізації. База даних складається з урахуванням характеристик об'єктів проектування, процесу проектування, діючих нормативів і довідкових даних. При створенні автоматизованих банків даних одним з основних є принцип інформаційної єдності, що полягає у використанні єдиної термінології, умовних позначок, символів, єдиних проблемно-орієнтованих мов, способів подання інформації, єдиної розмірності даних фізичних величин, що зберігаються в БД. Автоматизовані банки даних повинні бути гнучкими, надійними, наочними й економічними. Гнучкість полягає в можливості адаптації, нарощування і зміни засобів СУБД і структури БД. Реорганізація БД не повинна призводити до зміни прикладних програм. Для одночасного обслуговування користувачів повинен бути організований спільний доступ до даних. При використанні інтерактивних методів проектування

необхідно використовувати режим діалогу.

Усі версії СУБД повинні генеруватися відповідно до наявного комплексу ТЗаб САПР. Доступ до інформації БД повинен забезпечуватися користувачами різних рівнів.

Для надійності необхідно мати можливість відновлення інформації і ПЗ у випадку руйнування, забезпечувати стандартні реакції на помилковий запит. Для наочності інформація повинна подаватися в звичайній і зручній для користувача формі. Для забезпечення економічності необхідно виключити невиправдане дублювання даних, забезпечити автоматизацію збору статистичних даних про зміст і використання інформації банку для організації ефективного розподілу пам'яті, забезпечити наявність засобів для тиражування баз даних.

Лінгвістичне забезпечення (ЛЗ) САПР – сукупність мов проектування (МП), включаючи терміни і означення, правила формалізації природної мови і методи ущільнення і відновлення текстів, необхідних для виконання АП, поданих у заданій формі.

Методичне забезпечення (МТЗ) САПР – сукупність документів, що встановлюють склад і правила добору й експлуатації засобів забезпечення АП, необхідних для виконання АП. Відзначимо, що в деяких роботах і документах методичне забезпечення розуміється більш широко: як компоненти включає МЗ і ЛЗ.

Організаційне забезпечення (ОЗ) САПР – сукупність документів, що встановлюють склад проектної організації і її підрозділів, зв'язки між ними, їх функції, а також форму подання результату проектування і порядок розгляду проектних документів, необхідних для виконання АП. Компонентами ОЗ САПР є методичні і керувальні матеріали, положення, інструкції, накази й інші документи, що забезпечують взаємодію підрозділів проектної організації при створенні й експлуатації САПР.

В наш час при створенні ЕОМ п'ятого покоління з архітектурою, що перебудовується, здатних приймати рішення в умовах розпливчастості і невизначеності, перспективними є САПР, які дозволяють адаптуватися до зовнішніх умов проектування і які мають можливість налаштовуватися на заданий клас створюваних об'єктів.

Інтегрованою називають САПР, що має альтернативне ПЗ й ОЗ САПР і що дозволяє вибирати сукупність машинних програм відповідно до задачі чи класу об'єктів проектування.

Функціонування САПР – виконання проектування в САПР відповідно до заданого алгоритму проектування. Функціонування САПР повинно забезпечувати одержання проектних рішень, тобто проміжних чи кінцевих описів об'єкта проектування, необхідних для його закінчення. Результатом проектування в САПР вважають сукупність закінчених проектних рішень, що задовольняють ТЗ, і необхідних для створення об'єкта проектування.

Алгоритм функціонування САПР – сукупність розпоряджень, необхідних для функціонування САПР.

Керування САПР – сукупність впливів ззовні, передбачених алгоритмом функціонування САПР.

2.3 Класифікація САПР

У загальному значенні класифікація – система супідрядних понять, що подається часто у вигляді різних схем, таблиць і використовується як засіб для встановлення зв'язків між цими класами об'єктів, а також для точної орієнтації в різноманітті понять чи відповідних об'єктів. Класифікація фіксує місце об'єкта в системі, що вказує на його властивості. У зв'язку з цим, вона служить засобом збереження і пошуку інформації, що міститься в ній самій. Класифікація створює умови для розробки технічно обґрунтованих норм забезпечення процесу створення, функціонування і стандартизації в галузі САПР.

Системи автоматизованого проектування класифікуються:

1. за типом, різновидом і складністю об'єкта проектування;
2. за рівнем і комплексністю автоматизації проектування;
3. за характером і числом проектних документів, що випускаються;
4. за числом рівнів у структурі технічного забезпечення.

На цьому класифікація першого рівня закінчена. Розглянемо класифікацію другого ієрархічного рівня (рисунок 2.2).

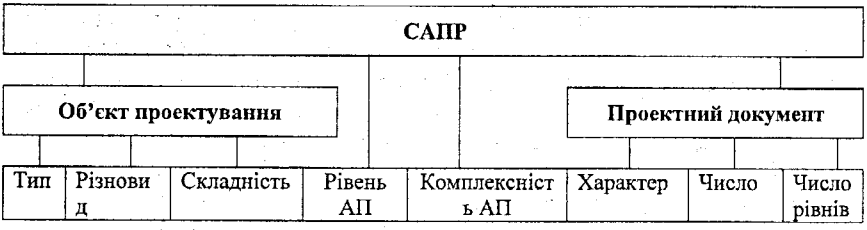


Рисунок 2.2 – Класифікація САПР

1. За *типом об'єкта проектування* розрізняють САПР:
   - виробів машинобудування і приладобудування;
   - технологічних процесів у машинобудуванні і приладобудуванні;
   - об'єктів будівництва;
   - організаційних систем.
2. За *різновидом об'єкта проектування* можна розрізняти САПР РЕА, САПР атомної станції, САПР ракетної системи і т.п.
3. За *складністю об'єкта проектування* розрізняють САПР:
   - прості, що містять до 10^2 складових частин;
   - середньої складності, що містять від 10^2 до 10^3 складових частин;
   - складні, що містять від 10^3 до 10^4 складових частин;

- дуже складні, що містять від 10^4 до 10^6 складових частин;
- надзвичайно складні, що містять 10^6 і більше складових частин.

Отже, САПР ТЕЗ ЄС ЕОМ відноситься до САПР простих об'єктів, а САПР багатопроцесорних обчислювальних систем на надвеликих інтегральних мікросхемах – до САПР дуже складних об'єктів.

4. За *рівнем автоматизації проектування* розрізняють САПР:

- низькоавтоматизовані, у яких число автоматизованих проектних процедур (АПП) складає 25% загального числа проектних процедур;
- середньоавтоматизовані, – від 25% до 50% загального числа проектних процедур;
- високоавтоматизовані від 50% до 75 %. У цих системах застосовують методи різноманітного оптимального проектування.

Наприклад, у гнучких автоматизованих виробництвах для ефективності результатів необхідно використовувати САПР середньо- і високоавтоматизованого проектування.

5. За *комплексністю САПР* класифікують:

- одноетапні, виконуючі один етап проектування з усіх установлених для об'єкта;
- багатоетапні, що виконують кілька етапів проектування з усіх установлених для об'єкта;
- комплексні, що виконують всі етапи проектування, установлені для об'єкта.

6. За *характером проектних документів*, що випускаються, розрізняють САПР:

- текстові, що створюють тільки текстові документи на паперовому листі;
- текстові і графічні, що створюють текстові і графічні документи на паперовому листі;
- на машинних носіях, що створюють документи на перфоносіях (перфокарти, перфострічки) і на магнітних носіях (магнітних стрічках, дисках і барабанах);
- на фотоносіях, що створюють документи на мікроплівках, фотошаблонах і т.п.;
- на двох типах носіїв; на всіх типах носіїв.

7. За *кількістю проектних документів*, що випускаються, розрізняють САПР:

- малої продуктивності;
- середньої продуктивності;
- високої продуктивності.

У різних галузях вводяться різні кількісні характеристики інформації, що визначають продуктивність САПР.

8. За *числом рівнів у структурі технічного забезпечення* розрізняють САПР:

- одноріВневі, побудовані на основі ЕОМ середнього чи високого класу зі штатним набором периферійних пристроїв, що може бути доповнений засобами обробки графічної інформації;
- дворівневі, побудовані на основі ЕОМ середнього чи високого класу й одного чи декількох АРМ, що включають міні-ЕОМ;
- трирівневі, побудовані на основі ЕОМ високого класу, АРМ і периферійного програмно-керованого обладнання.

Дана класифікація – не догма. Теорія автоматизації проектування безупинно розвивається. З'являються нові технічні і програмні засоби ЕОМ, комплексні САПР, тому існуючі схеми класифікації САПР будуть видозмінюватися й удосконалюватися.

2.4 Системний підхід до автоматизації проектування і принципи організації САПР

Розглянемо основні принципи організації САПР. Метою створення САПР є підвищення якості і техніко-економічного рівня проектованих об'єктів при їхньому створенні і застосуванні, підвищення продуктивності праці, скорочення термінів, зменшення вартості і трудомісткості проектування.

Очевидно, що створення САПР, у першу чергу, повинно бути обумовлено техніко-економічною доцільністю.

2.4.1 Структурна організація САПР

Складеними структурними частинами САПР є *підсистеми*, що володіють усіма властивостями систем і створюються як самостійні.

Підсистемою САПР називають виділену по деяких ознаках частину САПР, що забезпечує одержання закінчених проектних рішень.

За призначенням підсистеми САПР розділяють на *проектувальні* і *обслуговувальні:*

- До проектувальних відносять підсистеми, що виконують проектні процедури й операції, наприклад, підсистема логічного проектування, підсистема конструкторського проектування, підсистема технологічного проектування, підсистема проектування деталей і складальних одиниць і т.п.
- До обслуговувальних відносять підсистеми, призначені для підтримки працездатності підсистем, що проектують, наприклад підсистема інформаційного пошуку, підсистема документування, підсистема графічного відображення об'єктів проектування і т.п.

Стосовно об'єкта проектування розрізняють об'єктно-орієнтовані (*об'єктні*) і об'єктно-незалежні (*інваріантні*) підсистеми. До об'єктних відносять підсистеми, що виконують одну чи декілька проектних процедур

чи операцій, безпосередньо залежних від конкретного об'єкта проектування. До інваріантних відносять підсистеми, що виконують уніфіковані проектні процедури й операції, наприклад, функції відпрацьовування, що не залежать від особливостей проектованого об'єкта.

Підсистеми складаються з компонентів, об'єднаних загальною для даної підсистеми цільовою функцією, які забезпечують функціонування цієї підсистеми.

Структурна єдність підсистеми забезпечується зв'язками між компонентами САПР.

2.4.2 Принципи створення САПР

До принципів створення САПР відносять такі принципи:

- включення;
- системної єдності;
- розвитку;
- комплексності;
- інформаційної єдності;
- сумісності;
- стандартизації.

Принцип *включення* забезпечує розробку САПР на основі вимог, що дозволяють включати цю САПР у САПР більш високого рівня.

Принцип *системної єдності* полягає в тому, що при створенні, функціонуванні і розвитку САПР зв'язки між підсистемами повинні забезпечувати цілісність системи.

Відповідно до принципу *розвитку*, САПР повинна створюватися і функціонувати з урахуванням поповнення, удосконалення і відновлення підсистем і компонентів.

Принцип *комплексності* забезпечує взаємопов'язаність проектування елементів і всього об'єкта на всіх стадіях, дозволяє здійснювати узгодження і контроль характеристик елементів і об'єкта в цілому.

Принцип *інформаційної єдності* полягає у використанні в підсистемах, компонентах і засобах забезпечення САПР єдиних умовних позначень, термінів, символів, проблемно-орієнтованих мов, способів подання інформації, що відповідають прийнятим нормативним документам.

Відповідно до принципу *сумісності мови* символи, коди, інформаційні і технічні характеристики структурних зв'язків між підсистемами, засобами забезпечення і компонентами повинні забезпечувати спільне функціонування підсистем і зберігати відкриту структуру системи в цілому.

Принцип *стандартизації* полягає в проведенні уніфікації, типізації і стандартизації підсистем і компонентів, інваріантних до проектованих

об'єктів і галузевої специфіки, а також у встановленні правил з метою упорядкування діяльності в області створення і розвитку САПР.

Підсистеми повинні вводитися в дію і функціонувати незалежно від інших підсистем. Єдність загальносистемних вимог забезпечує проектна служба САПР.

2.4.3 Стадії створення САПР

Розрізняють зовнішнє і внутрішнє проектування.

До ***зовнішнього проектування*** відносяться такі стадії:

- *передпроектне дослідження* – проводиться обстеження проектної організації, оформлення технічного звіту, а також його узгодження і затвердження;
- *розробка, узгодження і затвердження технічного завдання* – виконуються спільно виконавцем і замовником.

До ***внутрішнього проектування*** відносяться такі стадії:

- *розробка технічної пропозиції* – вибирають і обґрунтовують оптимальний варіант САПР. Здійснюється розробка комплекту документації, а також узгодження і затвердження технічної пропозиції;
- *ескізний проект* – розробляють принципові рішення щодо створення САПР і документації, погоджують і затверджують ескізний проект;
- *технічний проект* – розробляють остаточні рішення зі створення САПР і документації, погоджують і затверджують технічний проект;
- *робочий проект* – розробляють робочу документацію з САПР, погоджують і затверджують робочий проект;
- *виготовлення, налагодження й випробування* – виготовляють і налагоджують компоненти САПР, виконують монтаж, налагодження й випробування комплексу засобів АП, здійснюють підготовку організації до введення системи в дію;
- *запровадження в дію* – проводять дослідження функціонування і приймальні випробування САПР.

Схематично процес створення САПР наведено на рисунку 2.3.

Роботи з підготовки організації до введення в дію САПР, підсистем і компонентів повинні вироблятися на всіх стадіях її створення. Перераховані стадії САПР не завжди є обов'язковими. Іноді вони відсутні чи поєднуються в одну чи декілька стадій, залежно від конкретних вимог до проектованої САПР.

Розглянемо докладніше основні стадії створення САПР.

1. До робіт на стадії *передпроектних досліджень* (ПД) відносять обстеження проектної організації, оцінку можливості створення САПР, збір даних, опис існуючих САПР і їх аналіз, збір пропозицій щодо

створення САПР, щодо складу підсистем, розробки компонентів САПР, формування технічних вимог до функцій і структури САПР, видів забезпечення, загальносистемних принципів створення САПР. Загальне керівництво ПД здійснює головний конструктор САПР. Після закінчення ПД оформляється і затверджується науково-технічний звіт.

Процес створення САПР							
Зовнішнє проектування		Внутрішнє проектування					
Передпроектні дослідження	Технічне завдання	Технічна пропозиція	Робочий проект	Ескізний проект	Виготовлення, наладка і випробування	Впровадження в дію	Технічний проект

Рисунок 2.3 – Схема процесу створення САПР

2. *Технічне завдання* (ТЗ) є вихідним і обов'язковим документом для створення, приймання чи здачі системи і повинно містити всі вихідні дані і вимоги, необхідні для створення САПР. Технічне завдання на створення САПР розробляє організація-розробник системи з урахуванням виконання перед проектних досліджень на основі вимог замовника. Воно узгоджується з організацією-користувачем САПР і, за необхідності, з іншими зацікавленими організаціями. Технічне завдання на створення САПР затверджує замовник. Змінюють і уточнюють затверджене ТЗ у процесі створення підсистеми за допомогою доповнень у порядку, встановленому в ТЗ.

Повне ТЗ на створення САПР ЕОА повинно містити такі розділи:

- *найменування й галузь застосування* – зазначено повне чи умовне найменування (індекс) САПР, коротка характеристика області її застосування;
- *підстава для створення* – зазначено повне найменування директивних документів, на підставі яких створюють САПР, організація, що затвердила ці документи, і дата затвердження, а також найменування чи умовна позначка теми, у рамках якої створюється САПР;
- *характеристика об'єкта проектування* – дані про призначення, склад, умови застосування розроблюваного об'єкта;
- *мета і призначення* – зазначено мету створення САПР, її призначення і критерій ефективності її функціонування;
- *характеристика процесу проектування* – наведено загальний опис процесу проектування, вимоги до вхідних і вихідних даних, а також

вимоги до поділу проектних процедур, які виконуються при автоматизованому і неавтоматизованому проектуванні;

- *вимоги до САПР* – наведено вимоги до САПР у цілому, до складу її підсистем, до включення в САПР створених підсистем і компонентів, вимоги до взаємозв'язку САПР з іншими системами, видів забезпечення і можливостей розвитку САПР;
- *техніко-економічні показники* – оцінено витрати на створення САПР, джерела економії, очікувана ефективність від застосування САПР, вимоги до техніко-економічних показників об'єкта, що будуть досягнуті в результаті функціонування САПР;
- *стадії и етапи* – встановлено необхідні стадії створення, черговість впровадження в дію САПР, етапи по стадіях, терміни виконання робіт і виконавці, перелік документації, пропонованої після закінчення стадії, обсяги робіт відповідно до діючих нормативно-технічних і методичних документів;
- *порядок випробувань і впровадження в дію* — визначено вимоги до проведення випробувань і запровадження в дію САПР і їхніх підсистем;
- *джерела розробки* – дано перелік науково-дослідних, дослідно-конструкторських і експериментальних робіт, нормативних документів, методичних матеріалів і т.п., використовуваних при створенні САПР;
- *додатки* – включають таблиці, схеми, розрахунки, бібліографію, лістинги програм, обґрунтування, докази і т.п.

Технічне завдання на створення САПР оформляють відповідно до загальних вимог щодо текстових документів.

3. Метою *технічної пропозиції* (ТП) є вибір раціональних варіантів САПР, що враховують вимоги ТЗ. При розробці ТП виконуються такі роботи:

- аналіз процесів автоматизованого проектування;
- виявлення можливих варіантів структури для САПР на основі розробки модулів, що реалізують процеси проектування;
- вибір раціональних варіантів (варіанта) структури САПР;
- техніко-економічне обґрунтування обраного варіанта;
- складання додаткових, порівняно з ТЗ, вимог до САПР;
- уточнення вимог до змісту робіт на наступних стадіях САПР.

Технічну пропозицію розробляє організація, що є головним розробником системи, її погоджують із замовником і з усіма зацікавленими організаціями. Технічна пропозиція на створення САПР повинна містити такі розділи:

- *загальні положення* – мета розробки ТП, найменування і дата затвердження ТЗ;
- *призначення й галузь застосування* – призначення, характеристика, галузь і умови застосування створюваної САПР;

- *опис об'єкта проектування в САПР* – основні складові елементи об'єктів проектування, їхній взаємозв'язок, схема розподілу;
- *опис процесу автоматизованого проектування* – результати аналізу процесу АП;
- *характеристика й аналіз варіантів структури САПР* – результати синтезу й аналізу структури САПР із виділенням підсистем і компонентів, зв'язків між ними, пропозиції щодо використання існуючих підсистем і компонентів САПР;
- *обґрунтування й опис обраного варіанта* – за результатами розгляду варіантів структури САПР влаштовується й описується обраний варіант САПР із указівкою взаємодії підсистем, компонентів і виконуваних ними функцій;
- *техніко-економічне обґрунтування* – основні техніко-економічні показники створюваної САПР;
- *пропозиції щодо змісту й організації робіт на наступних стадіях* – уточнені, порівняно з ТЗ, дані з черговості і змісту робіт.

Технічна пропозиція може мати, за необхідності, додаткові розділи і додатки. В останніх наводять схеми, описи, розрахунки, необхідні для якісного обґрунтування обраного варіанта САПР.

4. *Робочий проект* (РП) створюється на підставі технічного проекту організацією-розробником САПР.

На стадії РП проводять:

- розробку детальної структури САПР, її підсистем, взаємозв'язку з іншими системами і її уточнення;
- побудову алгоритмів і структурних схем автоматизованих процесів проектування;
- формування МЗ, ПЗ, ІЗ, ОЗ;
- розробку документації для монтажу, настроювання й експлуатації САПР;
- створення проектів програм і методик випробувань і дослідної експлуатації; оформлення і затвердження.

Результатом робіт на цій стадії є документ «Робочий проект» і комплект експлуатаційних документів.

Робочий проект містить:

- відомість РП;
- пояснювальну записку;
- специфікацію – перераховано підсистеми, специфікації видів забезпечення;
- документацію технічного забезпечення – специфікація, комплект конструкторських документів;
- документацію ІЗ – специфікація, опис бази даних, інструкція з її заповнення, інструкція з уведення масивів даних;

• документацію ПЗ – специфікація, тексти програм, опис програм, порядок і методика випробувань;

• документацію МЗ, МТЗ, ЛЗ – містить специфікацію, пояснювальну записку, опис мови, підсистеми керування САПР;

• документацію ОЗ – специфікація, програма підготовки фахівців-користувачів САПР, положення про службу САПР;

• програму і методику випробувань КЗАП – містить технічні дані, що підлягають перевірці при випробуванні компонентів САПР, порядок випробувань і методи їхнього контролю;

• програму і методику дослідного функціонування системи і підсистем – наведено дані, що забезпечують одержання і перевірку проектних рішень, виявлення причин збоїв, показників якості функціонування системи і підсистем;

• комплект експлуатаційних документів – містить МТЗ, ТЗ, ПЗ і відомість експлуатаційних документів.

Виготовлення, налагодження й випробування САПР проводять на підставі документації РП. Вони повинні забезпечувати безперебійну роботу підсистем діючої САПР.

5. На стадії *ескізного проекту* (ЕП) виконують:

- прийняття основних рішень щодо взаємодії САПР з іншими системами;
- прийняття основних технічних рішень за структурою підсистем САПР, розробку питань функціонування підсистем на рівні компонентів;
- опис вихідних даних і вимог на розробку мов проектування, алгоритмів, компонентів ІЗ до кожної із підсистем;
- оформлення і затвердження ЕП.

6. На стадії *виготовлення, налагодження й випробувань* виконують:

- підготовку організації до запровадження в дію САПР, включаючи навчання користувачів;
- виготовлення і налагодження компонентів ТЗ, ПЗ і ІЗ на машинних носіях;
- монтаж, налагодження й випробування КЗАП для САПР і всіх її підсистем;
- організацію робіт з оформлення і затвердження актів здачі компонентів і КЗАП.

7. Роботи на стадії *впровадження в дію* САПР призначені для забезпечення можливості промислового функціонування САПР і визначення техніко-економічних показників системи.

На цій стадії проводять дослідження функціонування, приймальні випробування, коректування, доробку і випуск експлуатаційних документів, промислову експлуатацію і розвиток САПР.

Дослідне функціонування САПР припускає перевірку роботоздатності і взаємодії підрозділів організації з підсистемами САПР, виявлення несправностей і відмов, визначення фактичних техніко-економічних показників системи і коректування документації, усунення дефектів, оформлення протоколу.

Приймальні випробування – це оцінка якості розробки КЗАП, перевірка готовності організації до промислового функціонування САПР, оцінка якості дослідного функціонування, перевірка документації на САПР. Робота закінчується впровадженням у дію системи, оформленням і затвердженням протоколу дослідного функціонування й акта приймання САПР у промислове функціонування.

Промислову експлуатацію виконує служба САПР. Проектна організація для визначення відповідності науково-технічному рівню проводить періодичні випробування САПР. При цьому оформляють висновок про відповідність підсистем і компонентів діючої САПР сучасним вимогам, визначають оцінку ефективності і якості продукції, отриманої за допомогою САПР, записують рекомендації щодо розробки чи модернізації підсистем і компонентів САПР, що перевіряється.

8. На стадії *технічного проекту* виконують:

- прийняття рішень щодо нового процесу проектування з забезпеченням взаємодії і сумісності автоматичних і автоматизованих процедур, одержання остаточної схеми функціонування САПР у цілому;
- розробку структури і складу підсистем САПР;
- одержання остаточної структури усіх видів забезпечення САПР;
- вибір математичних моделей об'єкта і проектування його елементів;
- розробку алгоритмів проектних операцій;
- розробку вимог до створення програм реалізації процедур проектування;
- розробку алгоритмів, мов проектування, компонентів ІЗ, формування загальносистемного програмного забезпечення;
- розрахунок продуктивності і вибір режимів роботи засобів технічного забезпечення;
- визначення вимог до подання вихідних даних, результатів проектування і проектних документів;
- оформлення і затвердження сукупності документів, що складають технічний проект.

Технічний проект містить у собі такі основні документи:

- відомість; пояснювальну записку;
- схеми процесів проектування, підсистем і засобів забезпечення, специфікації компонентів усіх видів забезпечення;

- кошторис витрат на створення САПР;
- ТЗ на розробку співвиконавцями з окремих компонент; розрахунок очікуваних техніко-економічних показників.

Розвиток САПР ведуть шляхом модернізації підсистем, компонентів і засобів забезпечення, а також введення в експлуатацію нових підсистем і компонентів.

Розглянемо основні функції і рекомендації зі структури служб САПР в організаціях. Основними функціями служби САПР є:

- вивчення, аналіз і узагальнення вітчизняного і закордонного досвіду створення і функціонування САПР;
- дослідження і розробка системних питань створення САПР у галузі, організація створення галузевої системи, розробка галузевих інваріантних підсистем і компонентів САПР для галузі;
- аналіз запровадження в дію і дослідне функціонування САПР, забезпечення розробки науково-технічних прогнозів розвитку САПР у галузі;
- формування і контроль виконання галузевих координаційних (перспективних і річних) планів НІР і ОКР зі створення САПР у галузі;
- організація впровадження типових систем і компонентів САПР;
- координація і контроль за ходом створення САПР у галузі;
- забезпечення проведення експертизи й узгодження технічної документації на створювану САПР;
- участь у роботі комісії з запровадження в дію САПР в організаціях галузі, участь у створенні конкретних систем САПР;
- забезпечення створення і розвитку галузевого фонду компонентів САПР;
- розробка галузевих стандартів, що керують документообігом в галузі створення САПР, функціонування і розвитку САПР;
- організація галузевих заходів щодо обміну досвідом і технічної інформації в області САПР;
- розробка пропозицій з питань підготовки кадрів з САПР.

Структура служби САПР у головній організації галузі визначається положенням про службу САПР, затверджуваним керівництвом відомства.

До складу служби САПР у загальному випадку входять підрозділи:

- керування створенням і розвитком САПР;
- загальносистемних розробок в області МТЗ, МЗ, ПЗ, ЛЗ, ІЗ і ТЗ САПР загальногалузевого застосування;
- галузевого фонду алгоритмів і програм (ГФАП) САПР;
- інформації з архівом технічної документації;
- розробки типових підсистем і компонентів САПР загальногалузевого призначення;
- технічної експлуатації окремих видів засобів забезпечення.

Порядок взаємовідносин служби САПР з іншими підрозділами визначає положення про службу САПР. Керівництво діяльністю служби САПР здійснює головний конструктор з САПР у галузі.

Основні функції служби САПР в організаціях такі:

- проведення передпроектних обстежень в організації;
- узгодження ТЗ на створення САПР;
- виділення в підрозділах-користувачах САПР групи фахівців, що забезпечують функціонування і розвиток підсистем і компонентів САПР у відповідно до спеціалізації підрозділів;
- забезпечення створення САПР;
- реалізація прив'язки типових підсистем і компонентів САПР до умов організацій;
- участь у розробці індивідуальних компонентів САПР з урахуванням специфіки конкретного об'єкта проектування;
- організація навчання і роботи фахівців-користувачів із КЗАП; забезпечення проведення приймальних випробувань САПР;
- організація виготовлення комплексу нестандартних компонентів САПР;
- організація робіт з реконструкції приміщень;
- здійснення підготовки і перенесення на машинні носії необхідної інформації для заповнення бази даних проектування;
- організація монтажних і налагоджувальних робіт, необхідних для функціонування САПР;
- здійснення планомірного розвитку САПР;
- забезпечення використання типових підсистем і компонентів САПР.

До складу служби САПР у загальному випадку входять такі підрозділи:

- організаційно-методичного забезпечення;
- розробки і розвитку окремих компонентів САПР з урахуванням специфіки підприємства;
- технічного забезпечення й експлуатації засобів обчислювальної техніки (ОТ);
- ведення архіву технічної документації САПР, засобів ІЗ, ПЗ на магнітних носіях, а також організації і диспетчеризації робіт.

Керівництво діяльністю служби САПР здійснює головний конструктор САПР в організації. Служба САПР забезпечує промислову експлуатацію САПР.

Розвиток САПР ведуть шляхом введення нових підсистем і компонентів в експлуатацію, а також на основі модернізації існуючих підсистем і компонентів.

Моніторна система (МС) САПР, тобто підсистема керування САПР, є обслуговуючою підсистемою САПР і призначена для організації й оптимізації керування процесом при виконанні проектних процедур і взаємодії підсистем САПР. Моніторна система у загальному випадку містить у собі компоненти математичного, програмного й інформаційного забезпечення.

До складу МС САПР входять процедурно-орієнтовані мови (метамови) проектування і керування процесом проектування, мови описів функцій МС, тобто розподілу ресурсів САПР, формування планових завдань і інше, мови опису структури даних і перетворення бази даних, програми, що забезпечують реалізацію функцій МС, бази даних МС.

Основні функції МС САПР:

- керування процесом реалізації проектних процедур і операцій;
- організація впливу підсистем САПР;
- інтерпретація мовних форм завдань на виконання проектних процедур і операцій;
- розподіл ресурсів САПР у процесі проектування;
- захист ресурсів системи і баз даних САПР від недозволеного доступу;
- забезпечення діалогових та інтерактивних режимів роботи при проектуванні в умовах одночасної роботи декількох підсистем САПР.

Програмне забезпечення МС САПР повинно функціонувати під керуванням операційної системи ЕОМ. Моніторна система повинна забезпечувати налаштування, а також контроль і відновлення процесу функціонування програм, мати властивості інформаційної сумісності з прикладними пакетами і підсистемами САПР на рівні керувальних параметрів, констант і інтерфейсів обміну інформацією.

2.4.4 Діалогові засоби САПР

Розглянемо загальні вимоги до *діалогових засобів* (ДЗ), що знаходять усе більшу вагу в сучасних САПР. До діалогових засобів САПР відносять засоби, що забезпечують пряму (інтерактивну) взаємодію користувача з КЗАП, здійснювану в реальному масштабі часу.

Діалоговий режим — це проектування з застосуванням ДЗ, при якому користувач, взаємодіючи з КЗАП, видає інструкції керування проектуванням. Діалогові засоби складають частину КЗАП і забезпечують переведення САПР із пакетного режиму в діалоговий.

Пакетний режим – це проектування за відсутності ДЗ і безпосереднього впливу користувача на процес проектування.

Залежно від виду подання даних ДЗ підрозділяють на символьні, графічні і змішані. До символьних відносять ДЗ, що забезпечують подання

даних у символьному (алфавітно-цифровому) вигляді, до графічних – ДЗ у графічному вигляді, а до змішаних – ДЗ як у графічному, так і в символьному вигляді. Залежно від режиму застосування ДЗ САПР підрозділяють на засоби автономного, неавтономного і змішаного застосування.

До складу ДЗ САПР входять: МЗ і ЛЗ, що містять інструкції і методики ведення діалогового режиму, мова діалогової взаємодії і мова внутрішнього подання даних:

- ТЗаб, до якого входять пристрої введення/виведення даних на електронних і інших типах дисплеїв, процесори ЕОМ, пристрої обміну даними;
- ПЗ, до складу якого входять програми діалогового введення і виведення даних, програми інтерпретатора мови діалогової взаємодії, програми керування процесом діалогу, програми формування результатів проектування, програми виведення результатів на дисплей і їхнє редагування, програми взаємодії з іншими обслуговувальними підсистемами САПР, програми реєстрації процесу діалогу і видачі користувачу довідок, інструкцій, повідомлень про помилки;
- ІЗ, до якого входять БД для процесу діалогу, БД для реєстрації і коректування результатів проектування, бібліотека типових графічних елементів, технічних даних і параметрів типових елементів проектування.

Рекомендується передбачати наявність в складі ДЗ засобів навчання і самонавчання користувача, а також адаптацію ДЗ стосовно можливостей користувача.

Математичне забезпечення ДЗ САПР повинно містити оригінальні і типові методи проектування. Лінгвістичне забезпечення ДЗ САПР базується на природних мовах, загальноприйнятих символьних і графічних образах мов, воно повинно бути інваріантним відповідно ІЗ ДЗ САПР.

Технічні ДЗ САПР забезпечують введення/виведення даних з використанням символьних і графічних образів природних мов. Технічні ДЗ будують на основі серійно виготовленого периферійного устаткування ЕОМ, що забезпечує введення/виведення даних.

Програмне забезпечення ДЗ САПР створюють з використанням базового ПЗ засобів обчислювальної техніки. Програмне забезпечення ДЗ підрозділяють на загальне і спеціалізоване.

Загальне ПЗ ДЗ забезпечує: введення і виведення інформації на діалоговий термінал; видачу користувачу довідок, інструкцій і повідомлень про помилки; реєстрацію процесу діалогу і керування ним; адаптацію структури діалогу з урахуванням особливостей конкретного користувача; керування базами даних і їхній захист; редагування і

маніпулювання даними; формально-логічний контроль діалогу; відновлення процесу діалогу.

Спеціалізоване ПЗ ДЗ забезпечує: трансляцію або інтерпретацію вхідних повідомлень; формування інформації користувачу про результати проектування; лексичний контроль вхідних повідомлень.

Програмне забезпечення ДЗ повинно мати модульну структуру й адаптуватися до зміни структури і засобів ведення діалогу. Інформаційне забезпечення ДЗ САПР повинно бути сумісним з ІЗ неділогових засобів САПР. У нього повинні входити БД для керування процесом діалогу і забезпечення режимів навчання і адаптації.

2.4.5 Принципи системного підходу

Розглянемо питання системного підходу при створенні ЕОМ. Вважається, що дослідження об'єктів проектування за допомогою їхніх математичних моделей складають основну суть системного підходу. Виділяють такі принципи системного підходу як:

- *ієрархічність* – кожна підсистема чи елемент може розглядатися як окрема система;
- *структурність* – можливість опису системи за допомогою комутаційних зв'язків між її елементами;
- *взаємозалежність* – прояв властивостей системи тільки при взаємодії з навколишнім середовищем;
- *множинність опису* – опис системи на основі безлічі математичних моделей;
- *цілісність досліджуваної системи* – вивчення властивостей цілої системи на основі аналізу і знання частин цього цілого.

В основі системного підходу лежить дослідження об'єкта як системи, спрямоване на пошук механізмів цілісності об'єктів і виявлення всіх його зв'язків. Системний підхід обґрунтовує загальну оптимізацію розробки, проектування, конструювання, виробництва, експлуатації об'єкта.

Одна з найважливіших задач системного підходу – *вибір виду, числа, рівня складності, форми подання математичних моделей*. У загальному випадку системний підхід при проектуванні – це облік усіх факторів, що впливають на процес створення об'єкта. Іншими словами, *системний підхід – це рішення технічної задачі для частини з урахуванням цілого.*

Наприклад, задача проектування ЕОМ не може в наш час розглядатися як локальна, вона повинна розглядатися з врахуванням усіх системних критеріїв, а також з урахуванням стану розвитку обчислювальної техніки. Як системні критерії при проектуванні ЕОМ використовують такі показники, як складність, живучість, надійність, вартість, ефективність, продуктивність і ін. Використання системних критеріїв дозволяє здійснювати комплексну оптимізацію при проектуванні ЕОМ.

Відомі *принципи системного підходу* в застосуванні до проектування електронно-обчислювальної апаратури (ЕОА) можна сформулювати в такий спосіб:

1. ЕОА, що складається з проектованих окремо оптимальним образом частин, не є, в загальному випадку, оптимальною (значення оптимуму може бути тільки відносним), тому повинна оптимізуватися в цілому як єдиний об'єкт із заданим цільовим призначенням;
2. ЕОА повинна оптимізуватися за критерієм, що відбиває ціль оптимізації;
3. ЕОА оптимізується в умовах кількісно визначених обмежень на оптимізовані параметри.

Вище були описані задачі синтезу. Задачі аналізу при проектуванні є задачами дослідження моделей створюваних об'єктів. Виділяють фізичні (макети, стенди, блоки і т.п.) і математичні моделі. Математичні моделі – це сукупність математичних об'єктів із заданими відносинами між ними. Математичні моделі бувають функціональні, структурні і комутаційні. Функціональні ММ відображають фізичні й інформаційні процеси, що відбуваються в модельованому об'єкті; структурні ММ – геометричні властивості об'єктів; комутаційні ММ – з'єднання в модельованих об'єктах. При проектуванні об'єкта, зазвичай, використовують сукупність описаних моделей. На кожному етапі проектування можуть застосовувати різні модифікації ММ.

Величини, що фігурують у ММ об'єктів, називають параметрами. Розрізняють параметри зовнішні, внутрішні і вихідні. Зовнішні параметри характеризують властивості зовнішнього, стосовно проектованого об'єкта, середовища, внутрішні параметри – властивості елементів об'єкта, а вихідні параметри – властивості самого об'єкта.

2.5 Роль і функція людини в САПР

Основні тенденції розвитку САПР визначають розвиток інтерактивних систем автоматизованого проектування. Центральне місце в таких системах займає діалог конструктора з ЕОМ. Організація діалогу забезпечується інформаційними, програмними і технічними засобами САПР.

При виборі технічних засобів САПР, що забезпечують інтерактивну взаємодію конструктора з процесом проектування, серед можливих альтернатив варто керуватися таким:

- Використовувана ЕОМ повинна дозволяти організовувати роботу певного числа користувачів (тобто задовольняти вимоги до продуктивності) у режимі розподілу часу.
- Використовуване термінальне обладнання повинно відповідати ергономічним вимогам і вимогам ефективності роботи.

До термінального обладнання можна віднести такі пристрої: пультову друкарську машинку, телетайп, алфавітно-цифровий і графічний дисплей, акустичні пристрої.

Застосування цих пристроїв обумовлено конкретним класом розв'язуваних задач у САПР. В наш час практично у всіх застосуваннях стає економічно невигідним використання телетайпів і терміналів з посимвольним введенням/виведенням інформації. Дисплеї зі сторінковою обробкою інформації дозволяють збільшити продуктивність роботи людини.

Застосування графічних дисплеїв йде в напрямку подання терміналів як автономних систем зі спеціальними операційними системами керування роботою окремих апаратних і програмних компонентів термінала і взаємозв'язком термінала з основною ЕОМ. Поява супермікро-ЕОМ, базису автономного комплексу, наблизило комплекси за своїми можливостями до міні-ЕОМ. Розвиток комплексів йде по двох напрямках розробки однопрограмних систем, розрахованих тільки на одного користувача, і мультипрограмних систем, розрахованих на одночасну роботу декількох користувачів, з реалізацією віртуальної пам'яті.

Для досягнення високої продуктивності систем, необхідної для одночасної роботи багатьох користувачів і з'єднання функціональних блоків системи, використовують різні канали, що поєднують процесор, оперативну пам'ять, матричний процесор і різні джерела інформації.

У графічних терміналах для керування функціонуванням усіх компонентів використовують мікропроцесори; для обробки графічної інформації необхідні спеціалізовані процесори; до термінала можуть підключатися пристрої пам'яті.

В цілому, тенденції розвитку терміналів можуть бути охарактеризовані як рух до багатофункціональних систем з розподіленою обробкою інформації, заснованих на широкому використанні ВІС.

Операційна система ОС ЄС ЕОМ версії 6.1 працює в режимі мультипрограмної обробки завдань. Система призначена для організації діалогових систем колективного доступу на базі ЄС ЕОМ. Її основні функції полягають у забезпеченні одночасної і незалежної роботи групи користувачів, що одержують доступ до ресурсів обчислювальної системи через алфавітно-цифрові дисплеї.

Структурний склад комплексу можна варіювати, залежно від рівня САПР, але навіть на одному рівні його конфігурація визначається тільки в умовах конкретної розробки на основі аналізу розв'язуваних задач.

Контрольні запитання

1. Взаємодія синтезу та аналізу при проектуванні обчислювальних

засобів.

2. Вимоги до САПР та засоби їх реалізації.
3. Структура САПР.
4. Мета, принципи створення і функції САПР.
5. Характеристика компонентів САПР.
6. Стисла характеристика сучасних САПР ЕОМ та РЕА.

3 СКЛАД МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ САПР

Математичне забезпечення САПР складається з математичних моделей об'єктів проектування, методів і алгоритмів виконання проектних операцій і процедур.

Основу математичного забезпечення САПР складає математичний апарат для моделювання, аналізування й оптимізації.

У математичному забезпеченні САПР можна виділити *спеціальну* (об'єктно-орієнтовану) частину, що значною мірою відображає специфіку об'єкта, проектування, фізичні й інформаційні особливості його функціонування і тісно пов'язані з конкретними ієрархічними рівнями (ця частина охоплює математичні моделі, методи й алгоритми їхнього одержання, методи й алгоритми одноваріантного аналізу, а також велику частину використовуваних алгоритмів синтезу), і *інваріантну* частину, що включає в себе методи й алгоритми, слабко пов'язані з особливостями математичних моделей і використовувані на багатьох ієрархічних рівнях (це методи й алгоритми різноманітного аналізу і параметричної оптимізації).

При постановці задач оптимізації в САПР необхідно перетворити фізичні уявлення про призначення і ступінь корисності об'єкта в математичне формулювання екстремальної задачі, тобто потрібно визначити мету оптимізації і поняття оптимальності. Мета виражається в критерії відносності – правилі переваги в порівнюваних варіантах. Основу такого критерію складає цільова функція $P(X)$. Аргументами цієї функції є керовані параметри, вектор яких позначимо через X.

Цільова функція повинна бути такою, щоб за її значеннями можна було визначити ступінь досягнення мети. Наприклад, кращий варіант повинен характеризуватися великим значенням $P(X)$, тоді оптимізація полягає в максимізації $P(X)$.

Крім цільової функції і переліку керованих параметрів, у постановку задачі оптимізації можуть входити обмеження типу рівностей $З(X) = 0$ і нерівностей $З(X) > 0$. Часним випадком обмежень типу нерівностей є прямі обмеження на керовані параметри, що мають вигляд $x_{min} < x_i < x._{мах}$, де x_{min}, $x_{мах}$ – гранично припустимі за фізичними, технологічними, чи іншими розуміннях, значення параметра x_i Зона простору керованих параметрів, у якій виконуються задані обмеження, називається *припустимою зоною.* При наявності обмежень задача оптимізації є задачею умовної оптимізації, при відсутності обмежень - задачею безумовної оптимізації. Значення, що характеризується найбільшим (найменшим) значенням $Г(X)$ при виконанні всіх обмежень, називається *умовним максимумом (умовним мінімумом).* При відсутності обмежень екстремум є *безумовним.*

Підсумкова задача оптимізації при проектуванні формулюється так: екстремувати цільову функцію $P(X)$ у припустимій зоні. У такій постановці

– це задача математичного програмування або одного з його розділів. Так, при лінійності цільової функції і функцій обмежень вона є задачею лінійного програмування; якщо хоча б одна з цих функцій нелінійна, – задачею нелінійного програмування. Точно так само за певних умов оптимізація може бути зведена до задач дискретного (частково-дискретного, цілочисельного, бівалентного) програмування. Методи розв'язання подібних задач оптимізації докладно викладені в роботах [26-28]. При розв'язанні багатьох задач синтезу дискретних пристроїв широко застосовуються логіко-комбінаторні методи. У найпростішій формі комбінаторний пошук зводиться до повного перебору варіантів. При цьому відомо заздалегідь, що серед них рішення задачі, що нас цікавить. Метод повного перебору застосовується тільки в простих ситуаціях. У загальному випадку пошук рішення повинен бути пов'язаний з істотним скороченням обсягу перебору, можливим лише на основі врахування додаткової інформації про задачі і властивості її розв'язків [17].

3.1 Вимоги до математичного забезпечення

Властивості МЗ мають істотний, а іноді і визначальний вплив на можливості і показники САПР.

При виборі і розробці моделей, методів і алгоритмів необхідно враховувати вимоги, пропоновані до МЗ в САПР.

1. *Універсальність.* Під універсальністю МЗ ми розуміємо можливість його застосування до широкого класу проектовних об'єктів. Високий ступінь універсальності МЗ потрібен для того, щоб САПР була застосовна до кожного чи більшості об'єктів, проектованих на підприємстві. Вона робить зручним використання ЕОМ, спрощуючи методику автоматизованого проектування.

Ступінь універсальності не має кількісної оцінки. Реалізуючи ту чи іншу модель або метод, розробник МЗ САПР повинен указати чіткі границі їхнього застосування.

2. *Алгоритмічна надійність.* Властивість компонента МЗ давати при його використанні в цих умовах правильні результати називається алгоритмічною надійністю. Ступінь універсальності характеризується заздалегідь обумовленими обмеженнями, а алгоритмічна надійність – обмеженнями, заздалегідь не виявленими, тобто не обумовленими.

Кількісною оцінкою алгоритмічної надійності служить імовірність одержання правильних результатів при дотриманні обумовлених обмежень на застосування методу. Якщо ця імовірність дорівнює одиниці чи близька до неї, то говорять, що метод алгоритмічно надійний.

Застосування алгоритмічно ненадійних методів у САПР є небажаним, хоча і припустимим у випадках, коли неправильні результати легко розпізнаються. Так як в САПР широко використовуються різні евристичні методи й алгоритми, необхідно досліджувати їхню

алгоритмічну надійність; якщо остання недостатня, варто вжити заходів чи до її підвищення, чи відмовитися від застосування методу.

З проблемою алгоритмічної надійності тісно пов'язана проблема обумовленості математичних моделей і задач. Про погану обумовленість говорять у випадках, коли малі похибкі вихідних даних призводять до великих похибок результатів. На кожному етапі обчислень є свої проміжні вихідні дані і результати, свої джерела похибок. При поганій обумовленості похибки можуть різко зростати. Це призводить як до зниження точності, так і до збільшення витрат машинного часу. Для аналізу й оптимізації об'єктів з погано обумовленими моделями необхідно використовувати спеціальні методи з підвищеною алгоритмічною надійністю.

3. *Точність*. Для більшості компонентів МЗ важливою властивістю є точність, обумовлена ступенем збігу розрахункових і правильних результатів. Алгоритмічно надійні методи можуть давати різну точність. І лише у випадках, коли точність виявляється гірше гранично допустимих значень або рішення взагалі неможливо одержати, говорять не про точність, а про алгоритмічну надійність.

В більшості випадків розв'язання проектних задач має такі особливості:

- спільне використання багатьох компонентів МЗ, що утруднює визначення внеску кожного з них у загальну похибку;
- результатом розв'язання є значення не окремого, а багатьох параметрів (тобто має місце векторний характер результатів).

У зв'язку з цим оцінка точності здійснюється за допомогою спеціальних обчислювальних експериментів, в яких створюються умови для окремої оцінки похибок, внесених математичними моделями елементів, алгоритмами аналізу й оптимізації. У цих експериментах використовуються спеціальні задачі, які називаються *тестовими*.

4. *Витрати машинного часу*. Універсальні моделі і методи характеризуються порівняно великим обсягом обчислень, що зростають зі збільшенням розмірності задач. Тому, при розв'язанні більшості задач, у САПР витрати машинного часу $T_м$ значні. Звичайно саме $T_м$ є головним обмежувальним фактором при спробах підвищити складність проектованих на ЕОМ об'єктів і старанність їхнього дослідження. Тому вимога економічності за $T_м$ – одна з основних вимог до МЗ САПР.

При використанні в САПР багатопроцесорних обчислювальних систем зменшити час розрахунків можна за допомогою паралельних обчислень. У зв'язку з цим одним з показників економічності МЗ є його пристосованість до розпаралелення обчислювального процесу.

Вимоги до високого ступеня універсальності, алгоритмічної надійності, точності, з одного боку, і малих витрат машинного часу, з іншого, суперечливі. Саме тому будь-який конкретний компонент МЗ,

відбиваючи певний компроміс у задоволенні цих вимог, може бути ефективним при розв'язанні однієї задачі і малоефективним при розв'язанні іншої. Тому в САПР доцільно мати бібліотеки з наборами моделей і методів, що перекривають потреби всіх користувачів САПР. Кожен компонент МЗ визначеного цільового призначення повинен бути поданий кількома різновидами, що забезпечують різний ступінь компромісного задоволення суперечливих вимог.

5. *Використовувана пам'ять*. Витрати пам'яті є другим, після витрат машинного часу, показником економічності МЗ. Вони визначаються довжиною програми й обсягом використовуваних масивів даних.

Незважаючи на значне збільшення об'єму оперативної пам'яті в сучасних ЕОМ, вимога економічності її витрат залишається актуальною. Це, зокрема, пов'язано з тим, що в мультипрограмному режимі функціонування ЕОМ задача з запитом великого об'єму пам'яті одержує більш низький пріоритет, і, в результаті, час її перебування в системі збільшується.

Для подолання труднощів, що виникають, можна використовувати зовнішню пам'ять. Однак часті обміни даними між оперативною і зовнішньою пам'яттю можуть призвести до зайвого збільшення часу виконання операції. Тому при великих обсягах програм і масивів оброблюваної інформації доцільно використовувати МЗ, що допускає побудову оверлейних програмних структур і реалізує принципи діакоптичної обробки інформації.

Значно вплинуло на розвиток МЗ САПР прагнення підвищити економічність використовуваних моделей і методів. Це досягається як розробкою економічних моделей і алгоритмів, що мають особистий характер, так і вдосконаленням і використанням загальних принципів створення МЗ, ефективного за витратами машинного часу і пам'яті. До таких принципів відносяться використання розрідженості матриць, дослідження складних систем частинами (*діакоптичні* методи дослідження), макромоделювання і раціональне використання здібностей людини в інтерактивних процедурах.

Врахування розрідженості матриць дозволяє в багатьох алгоритмах, у яких використовуються операції над матрицями, домогтися істотного скорочення витрат машинних часу і пам'яті. Він заснований на збереженні в ЕОМ тільки ненульових елементів матриць і виконанні арифметичних дій тільки над ними. У задачах проектування найчастіше функціонують сильно розріджені матриці (тобто з великим числом нульових елементів). Якщо враховувати це, то витрати машинного часу і пам'яті можна зробити лінійно залежними від показників складності аналізованого об'єкта, у той час як без обліку розрідженості ці залежності, зазвичай, мають квадратичний, чи навіть кубічний характер.

Відмінність діакоптичного підходу до проектування від *блочно-ієрархічного* полягає в тому, що діакоптика заснована на використанні структурних особливостей схем, а не на прийнятті будь-яких допущень, що спрощують. Тут здійснюється розчленовування математичних моделей на частини, досліджувані самостійно. Це дозволяє впорядкувати і мінімізувати кількість обмінів інформацією між оперативною і зовнішньою пам'яттю при аналізуванні складних схем, а також вибирати для дослідження кожної частини найбільш вигідні режими.

Раціональне використання евристичних здібностей людини в інтерактивних процедурах дозволяє інженеру втручатися в хід обчислень і вибирати найбільш перспективні продовження на основі евристичних оцінок. Це вигідно у всіх тих проектних процедурах, у яких проходження тільки формальним критерієм вибору подальших дій пов'язано з надмірними витратами машинного часу.

3.2 Математичні моделі

Знання особливостей математичних моделей, методів і алгоритмів розв'язання проектних задач необхідно інженеру для правильного формулювання вихідних даних і інтерпретації одержаних результатів. При автоматизації проектування специфіка створюваних об'єктів знаходить висвітлення, насамперед, у їхніх *математичних моделях*.

Математична модель (ММ) технічного об'єкта є сукупністю математичних об'єктів (чисел, змінних, матриць, множин і т.п.) і відносин між ними, які адекватно відображають властивості технічного об'єкта, які цікавлять інженера, що розробляє цей об'єкт.

Для складання математичних моделей можна використовувати будь-які математичні засоби – диференціальні й інтегральні рівняння, теорію множин, теорію графів, булеву алгебру, теорію автоматів і т.п. Процес складання математичної моделі називається *математичним моделюванням*. Це самий загальний і найуживаніший метод досліджень.

Вище вже говорилося про блочно-ієрархічне подання об'єктів проектування. Воно природним образом знаходить своє відображення й у способах побудови математичних моделей. На кожному ієрархічному етапі і рівні використовуються свої математичні моделі. Поєднання рівнів і етапів з метою організації загального процесу проектування вимагає узгодження цих моделей.

3.2.1 Класифікація математичних моделей

Виконання проектних операцій і процедур у САПР засновано на оперуванні ММ. З їхньою допомогою прогнозуються характеристики й оцінюються можливості запропонованих варіантів схем і конструкцій, перевіряється їхня відповідність пропонованим вимогам, проводиться оптимізація параметрів, розробляється технічна документація і т.п.

1. *Функціональні моделі.* У проектних процедурах, пов'язаних з функціональним аспектом проектування, як правило, використовуються ММ, які відбивають закономірності процесів функціонування об'єктів. Такі моделі називають функціональними. Типова функціональна модель – це система рівнянь, що описують електричні, теплові, механічні процеси або процеси перетворення інформації.

2. *Структурні моделі.* В той самий час у процедурах, які відносяться до конструкторського аспекту проектування, переважає використання математичних моделей, які відбивають лише структурні властивості об'єкта, наприклад його геометричну форму, розміри, взаємне розташування елементів у просторі. Такі моделі називають структурними. Структурні моделі найчастіше зображаються у вигляді графів, матриць, списків і т.п. Вони визначають взаємне розташування елементів у просторі, наявність безпосередніх зв'язків між ними у вигляді каналів, провідників і т.п. Такі моделі, зазвичай, використовують у випадках, коли задачі структурного синтезу вдається ставити і розв'язувати, абстрагуючись від особливостей фізичних процесів в об'єкті

Оскільки структурні і функціональні властивості об'єктів в значній мірі взаємозалежні, більшість проектних процедур вимагають моделей з відображенням особливостей як структури об'єкта, так і характеру фізичних чи інформаційних процесів, що відбуваються в ньому.

3. *Мікро-, макро- і метамоделі.* Залежно від складності об'єкта при його проектуванні використовують більшу чи меншу кількість рівнів абстракції. Об'єднання рівнів, споріднених за характером використовуваного математичного апарата, приводить до утворення трьох укрупнених рівнів (мікро-, макро- і метарівня) в ієрархії функціональних моделей для більшості проектовних складних об'єктів.

На *мікрорівні* використовують математичні моделі, що описують фізичний стан і процеси в суцільних середовищах. Для моделювання застосовують апарат рівнянь математичної фізики. Прикладами таких рівнянь є диференціальні рівняння в частинних похідних – рівняння електродинаміки, теплопровідності, пружності, газової динаміки.

На *макрорівні* здійснюється дискретизація просторів з виділенням елементів окремих деталей, дискретних електрорадіоелементів, ділянок напівпровідникових кристалів. При цьому з числа незалежних змінних виключають просторові координати. Функціональні моделі на макрорівні – це системи алгебраїчних чи звичайних диференціальних рівнянь, для їхнього одержання і розв'язання використовують відповідні чисельні методи.

Макромоделювання лежить в основі напрямку, пов'язаного з раціональним вибором математичних моделей елементів при побудові математичної моделі системи. Воно реалізує можливість використання при аналізі того самого об'єкта декількох моделей, що розрізняються

складністю, точністю і повнотою відображення властивостей об'єкта, трудомісткістю обчислень і т.п. Кожна з моделей відповідає певній розбивці системи на елементи і вибору певних моделей отриманих елементів. Найдетальніша розбивка в рамках даного ієрархічного рівня призводить до одержання повної математичної моделі, яка характеризується високою точністю й, у той самий час, великим обсягом обчислень, що вимагаються. Підвищити економічність можна поділом системи на більші блоки з використанням для них спрощених математичних моделей – *макромоделей.*

На *метарівні*, за допомогою подальшого абстрагування від характеру фізичних процесів, вдається одержати прийнятний за складністю опис інформаційних процесів, що протікають у проектованих об'єктах. На метарівні для моделювання аналогової РЕА широко застосовують апарат аналізу систем автоматичного керування, а для моделювання цифрової РЕА – математичну логіку, теорію кінцевих автоматів, теорію масового обслуговування. Математичні моделі на метарівні – системи звичайних диференціальних рівнянь, системи логічних рівнянь, імітаційні моделі систем масового обслуговування.

3.2.2 Форми подання моделей

Для представлення моделей використовують такі основні форми:

Інваріантна форма – запис співвідношень моделі за допомогою традиційної математичної мови незалежно від методу розв'язання рівнянь моделі.

Алгоритмічна форма – запис співвідношень моделі й обраного чисельного методу рішення у формі алгоритму.

Аналітична форма – запис моделі у вигляді результату аналітичного розв'язання вихідних рівнянь моделі; зазвичай моделі в аналітичній формі – це явні вираження вихідних параметрів як функцій внутрішніх і зовнішніх параметрів. Ці моделі характеризуються високою економічністю, але в той самий час можуть бути отримані лише в окремих випадках, як правило, при прийнятті істотних допущень і обмежень, що зменшують точність і звужують зону адекватності моделі.

Схематична форма, яку називають також *графічною формою*, – подання моделі деякою графічною мовою, наприклад, мовою графів, еквівалентних схем, діаграм і т.п. Графічні форми зручні для сприйняття людиною. Використання таких форм можливо при наявності правил однозначного тлумачення елементів креслень та їхнього перекладу на мову інваріантних чи алгоритмічних форм.

Моделі в алгоритмічній і аналітичній формах називають, відповідно, *алгоритмічними* й *аналітичними.*

Серед алгоритмічних моделей важливий клас утворюють *імітаційні моделі*, призначені для імітації фізичних чи інформаційних процесів в об'єкті при заданні різних залежностей вхідних впливів від часу. Власне

імітацію названих процесів називають *імітаційним моделюванням*. Результат імітаційного моделювання – залежності фазових змінних в обраних елементах системи від часу. Прикладами імітаційних моделей є моделі електронних схем у вигляді систем звичайних диференціальних рівнянь чи моделі систем масового обслуговування, призначені для імітації процесів проходження заявок через систему.

3.2.3 Вимоги до математичних моделей

Основними вимогами для математичних моделей є вимоги адекватності, універсальності й економічності.

Адекватність. Модель вважається адекватною, якщо відображає задані властивості об'єкта з прийнятною точністю. Точність визначається як ступінь збігу значень вихідних параметрів моделі й об'єкта.

Універсальність. При визначенні області адекватності (ОА) необхідно вибрати сукупність зовнішніх параметрів і сукупність вихідних параметрів y_i, що відображають властивості моделі. Збільшення числа зовнішніх факторів, що враховуються, розширює застосування моделі.

Якщо адекватність характеризується положенням і розмірами ОА, то універсальність моделі визначається кількістю і складом зовнішніх і вихідних параметрів, що враховуються в моделі.

Економічність. Економічність моделі характеризується витратами обчислювальних ресурсів для її реалізації, а саме, витратами машинного часу $T_м$ і пам'яті $П_м$.

3.2.4 Методи створення моделей елементів

Для створення математичних моделей використовуються неформальні і формальні методи.

Неформальні методи використовуються на різних ієрархічних рівнях для одержання математичних моделей елементів. Вони включають вивчення закономірностей процесів і явищ, пов'язаних з модельованим об'єктом, виділення істотних факторів, прийняття різного роду допущень і їхнє обґрунтування, математичну інтерпретацію наявних зв'язків і т.п. Використання неформальних методів можливо для побудови *теоретичних* і *експериментальних* математичних моделей.

Теоретичні методи засновані на вивченні фізичних закономірностей процесів, що протікають в об'єкті, визначенні відповідного цим закономірностям математичного опису, обґрунтуванні і прийнятті припущень, що спрощують виконання необхідних викладень і приведенні результату до прийнятої форми подання моделі.

Експериментальні методи засновані на використанні зовнішніх проявів властивостей об'єкта, що фіксуються під час експлуатації однотипних об'єктів чи при проведенні цілеспрямованих експериментів.

Розв'язання задач моделювання елементів полегшується завдяки тому, що для побудови більшості технічних об'єктів використовуються

типові компоненти. Тому розробка математичних моделей нових елементів здійснюється порівняно рідко. Один раз створені моделі потім багаторазово використовують при розробці різних систем, наприклад, моделей стандартних функціональних вузлів цифрової техніки (лічильників, дешифраторів), транзисторів, діодів і ін.

Формальні методи використовуються для одержання математичних моделей систем при відомих математичних моделях елементів.

У загальному випадку процедура одержання математичних моделей включає такі операції:

1. Вибір властивостей об'єкта, які підлягають відображенню в моделі. Цей вибір заснований на аналізі можливих застосувань моделі і визначає ступінь її універсальності.

2. Одержання вихідної інформації про обрані властивості об'єкта. Джерелами зведень можуть бути досвід і знання інженера, що розробляє модель, науково-технічна література, насамперед довідкова, опис прототипів – наявних математичних моделей, близьких за своїми властивостями до досліджуваного об'єкта, результати експериментального вимірювання параметрів і т п.

3. Синтез структури математичної моделі, під якою розуміється найзагальніший вигляд математичних співвідношень без конкретизації числових значень параметрів, що фігурують у них.

4. Розрахунок числових значень параметрів моделі. Ця задача ставиться як мінімізація похибок моделі заданої структури.

5. Оцінка точності й адекватності моделі.

При розробці математичної моделі операції 2÷5 можуть виконуватися багаторазово в процесі послідовних наближень до бажаного результату.

При виборі і розробці моделей, методів і алгоритмів, що складають МЗ, необхідно враховувати вимоги, що істотно, а іноді й суттєво впливають на можливості і показники САПР. До головних із них відносяться: універсальність, алгоритмічна надійність, точність, дотримання обмежень на витрати машинного часу і пам'яті.

3.3 Системне проектування

У підсистемі структурного проектування процедури одержання математичної моделі об'єкта й імітації його функціонування на цій моделі за допомогою програмної реалізації на ЕОМ називають *імітаційним моделюванням*.

Обчислювальні системи на системному рівні, як правило, розглядаються як *системи масового обслуговування* (СМО), а аналіз обчислювальних систем є статистичним.

3.3.1 Імітаційні моделі обчислювальних систем

Модель джерела вхідного потоку заявок є алгоритмом, за яким обчислюються моменти появи заявок на обслуговування. Джерела можуть бути незалежними і залежними. Модель незалежного джерела найчастіше реалізує алгоритм створення значень випадкової величини, розподіленої за заданим законом. Ця випадкова величина є проміжком часу між появами двох сусідніх заявок. У залежних джерелах заявка на виході з'являється при надходженні на вхід деякої іншої заявки, яка називається синхронізувальною. Кожне джерело виробляє заявки одного типу з визначеними пріоритетами.

Ресурси обчислювальної системи поділяються на *пристрої* і *пам'ять*. Пристрій може обслуговувати в кожен момент часу одну заявку, а пам'ять – кілька заявок.

Модель пристрою – це алгоритм генерації значень інтервалів обслуговування.

Найчастіше це алгоритм генерації значень випадкової величини, розподіленої за заданим законом. У моделі для кожного типу заявок може бути встановлений свій закон розподілу і його числові параметри. Крім того, модель пристрою відображає задану дисципліну обслуговування – до моделі входить алгоритм, що керує чергами відповідно до дисципліни обслуговування і пріоритетами заявок, що надійшли.

Модель пам'яті – це алгоритм для визначення об'єм пам'яті, що вимагається для обслуговування заявки.

Об'єм пам'яті визначається як генерація випадкової величини, причому закон розподілу і його параметри залежать від типу заявки. Параметрами пам'яті є її загальна ємність і дисципліна обслуговування. Заявка, що надійшла в пам'ять, займає обчислений об'єм і продовжує рух у системі аж до зустрічі зі спеціальним елементом, що називається елементом звільнення пам'яті.

Зв'язок джерел і області адекватності значною мірою визначається програмними засобами імітаційної системи. Саме вони визначають маршрути заявок між процесорами, пам'яттю та зовнішніми пристроями. Для імітації зв'язків між областями адекватності в моделі використовуються спеціальні оператори (елементи), які називаються *вузлами*.

Вузли можуть бути декількох типів:

- вузли одного типу служать для спрямування заявок тим чи іншим маршрутом, залежно від типу заявки, чи виконання деяких умов;
- вузли іншого типу – для поділу заявок на частини чи їхнє об'єднання, а також вузли для зміни параметрів заявок.

Таким чином, імітаційна модель є алгоритмом, що складається з упорядкованих звертань до моделей елементів – джерел, пристроїв,

пам'яті, вузлів. Послідовність звертань визначається властивостями аналізованої обчислювальної системи і режимом її функціонування. У процесі імітації в моделі відбуваються зміни дискретного часу. Час змінюється після того, як закінчено імітацію чергової групи подій, що відносяться до діючого моменту часу t_i. Зміна полягає в збільшенні часу на інтервал Δt, що відокремлює розглянуту групу події від будь-яких найближчих за часом подій.

Моменти настання подій, як це ясно з вищевикладеного, визначаються при звертаннях до моделей джерел і пристроїв. У процесі імітації відбувається нагромадження даних, таких як кількості заявок, що вийшли із системи обслугованими (не обслугованими), минуле обслуговування в кожному із пристроїв, підсумовування часу зайнятого стану кожного пристрою, обчислення середніх значень використовуваного об'єму пам'яті, довжин черг і т.п.

Процес імітації закінчується, коли поточний час t перевищить заданий відрізок T_k часу чи коли вхідні джерела згенерують задану кількість заявок. Після цього підраховуються вихідні параметри за тими даними, що були накопичені в процесі імітації.

Для аналізу методом імітаційного моделювання необхідні порівняно великі витрати машинного часу. Однак універсальність методу робить його основним інструментом аналізу при проектуванні обчислювальних систем.

3.4 Функціонально-логічне проектування

Задачі функціонально-логічного проектування – *розробка функціональних і принципових схем*. Основні задачі даного рівня, що розв'язуються на ЕОМ,-це задачі аналізу.

3.4.1 Моделювання функціональних схем цифрової РЕА

Для моделювання функціональних схем цифрової РЕА характерні такі особливості:

1. Стан елементів схем характеризується фазовими змінними одного типу, що визначають збережену чи передану інформацію, фізична природа цієї фазової змінної (напруга чи струм) не конкретизується.

2. Фазові змінні, що відображають інформаційний стан елементів, доцільно подавати в дискретній формі, оскільки інформація має цифрову форму. Звичайно фазові змінні можуть приймати значення з множини $\{0,1\}$, тоді вони називаються двійковими або *булевими*. У ряді випадків для аналізу функціональних схем зручно використовувати трійкові і п'ятіркові змінні.

3. Аналізування функціональних схем відбувається в дискретні моменти часу. Вісь часу поділяється на такти тривалістю Т. Зміна значення хоча б однієї фазової змінної називається *подією*. У більшості випадків

вважається, що такі зміни відбуваються миттєво. Якщо деяка подія відбулася між моментами часу $t + nT$ і $t + (n+1)T$, то в моделі ця подія відноситься до моменту часу $t + (n+1)T$. Часто використовується відносний час, що виражається в кількостях тактів (T- відношення абсолютного часу до T).

Розглянемо основні типи моделей функціональних схем цифрової РЕА.

Модель елемента функціональної схеми в загальному випадку задається системою рівнянь у вигляді:

$$\begin{cases} Y = \Psi(X, A); \\ A' = \varphi(X, A), \end{cases} \qquad (3.1)$$

де X – вектор вхідних змінних елемента;

A і A' – вектори внутрішніх змінних, що характеризують стан елемента;

Y – вектор вихідних змінних.

Якщо елементи векторів X відносяться до моменту відносного часу t, то елементи y_i вектора V – до моменту часу $t + k_j$, причому затримки k_j для різних j можуть бути різними. Також затриманими в часі є елементи вектора A'.

Модель елемента у вигляді (3.1) має такі особливості:

- змінні Y, X і A можуть бути не тільки двійковими, але і трійковими, п'ятірковими й ін.;
- враховуються тимчасові затримки в зміні стану елемента й у появі вихідної реакції на вхідне порушення, причому величини затримок k_j для конкретного типу елемента можуть прийматися постійними чи визначатися як значення випадкових величин, чи обчислюватися як значення функцій зовнішніх параметрів.

В окремому випадку одновихідного комбінаційного елемента модель (3.1) набуває вигляду:

$$y(t + k) = \Psi(X(t)),$$

де Ψ – логічна функція.

Математична модель функціональної схеми – сукупність математичних моделей елементів, у яких виконане ототожнення фазових змінних відносно виходів елементів, що з'єднуються.

Для запису математичної моделі функціональної схеми в загальному випадку зручно ввести такі позначення: U – вектор вхідних для схеми величин; V і V' – вектори вихідних і внутрішніх змінних всіх елементів схеми для поточного t і майбутнього t' моментів відносного часу відповідно. Для елемента v_j' вектора V' момент майбутнього часу t'

визначається як $t + k_j$ тобто значення затримок k_j можуть бути різними для різних змінних.

Вихідні змінні функціональної схеми є вихідними змінними деяких елементів і, отже, входять до векторів V і V'. Тоді математична модель функціональної схеми виражається системою рівнянь вигляду:

$$V' = F(V, U) \tag{3.2}$$

з дискретними змінними V, V', U, в якій F – оператор перетворення дискретних змінних.

Модель (3.2) називається *асинхронною* моделлю функціональної схеми. Якщо всі затримки $k_j \neq 0$, то асинхронна модель (3.2) є сукупністю рекурентних співвідношень, які дозволяють при відомих значеннях змінних V у q в початкових точках тимчасового діапазону і заданому законі зміни вхідних змінних $U(t)$ обчислити значення вектора V у всьому тимчасовому діапазоні, тобто, побудувати тимчасові діаграми роботи пристрою. Тут $q = \max\limits_{j \in [1:n]} k_j$, n – розмірність вектора V. Асинхронну модель можна розглядати як дискретний аналог динамічної моделі з неперервними змінними.

Асинхронні моделі універсальні, тому що можуть застосовуватися для аналізування як асинхронних, так і синхронних схем, дослідження перехідних процесів і сталих станів схем. Однак використання асинхронних моделей супроводжується великими витратами машинного часу. Скоротити ці витрати можна при аналізуванні синхронних схем, тобто схем, у яких передача інформації між регістрами дозволяється тільки при наявності спеціальних синхронізувальних імпульсів. У цих схемах, як правило, період синхроімпульсів T_c вибирається за такої умови, щоб перехідні процеси в схемах міжрегістрових пересилань відбувалися за період T_c.

Якщо встановлено, що ця умова виконується, то роботоздатність схем може бути перевірена шляхом аналізу сталих режимів. З цією метою використовуються економічніші синхронні моделі схем. У них математичні моделі елементів – це окремий випадок (3.1), що існує при $k_j = 0$. Тоді маємо $V' = V$ і синхронна модель функціональної схеми виходить з асинхронної моделі (3.2) у вигляді системи рівнянь, що називаються логічними рівняннями вигляду:

$$V = F(V, U), \tag{3.3}$$

у яких час не фігурує. Отримана система рівнянь є дискретним аналогом неперервних статичних моделей, що виражаються системами кінцевих рівнянь. Розв'язання дискретних рівнянь (3.3), як і їхніх неперервних аналогів, звичайно здійснюється ітераційними методами. Результат розв'язання – стале значення вектора змінних V на даному періоді синхроімпульсів.

Синхронні моделі використовуються для перевірки коректності з'єднань елементів у функціональній схемі, для виявлення ризиків збою, при проектуванні тестів.

3.5 Схемотехнічне проектування

Головні задачі схемотехнічного рівня пов'язані з *проектуванням принципових електричних схем*, реалізованих у кристалах інтегральних схем. Як і на інших ієрархічних рівнях проектування РЕА, при схемотехнічному проектуванні основними задачами, які розв'язуються за допомогою ЕОМ, є задачі аналізу.

3.5.1 Математичні моделі елементів електронних схем

Форма подання моделей. Елементи електронних схем розглядаються як окремі компоненти (транзистори, резистори, трансформатори і т.п.), так і деякі підсхеми – функціональні вузли (підсилювачі, логічні елементи, детектори, формувачі і т.п.), що є типовими для проектованої радіоелектронної апаратури.

Математичні моделі функціональних вузлів, розглянутих як елементи, повинні бути макромоделями.

До моделей елементів висуваються ті самі вимоги, що і до інших частин математичного забезпечення САПР (вимоги універсальності, точності й економічності).

Через те, що вимоги універсальності і точності, з одного боку, і економічності, з іншого, суперечливі, для більшості компонентів не існує моделей, що виявилися б найкращими в будь-яких умовах застосування. Тому в сучасних програмах аналізу, як правило, створюються бібліотеки моделей, для кожного компонента є декілька моделей, що відрізняються ступенем задоволення суперечливих вимог.

Є декілька форм подання моделей елементів:

1. *Схемна форма.* Модель подається у вигляді докладної еквівалентної схеми, що складається з двополюсників, з зазначенням компонентного рівняння для кожної гілки схеми, що є наочнішим для інженера-схемотехніка.

При формуванні ММ системи із схемних моделей елементів складається загальна еквівалентна схема, для якої формуються рівняння табличним методом, методом змінних станів чи методом вузлових потенціалів. Недолік такого методу формування ММ системи – значні витрати машинної пам'яті.

2. Інваріантна форма. Модель подається у вигляді системи рівнянь безпосередньо до методу їхніх чисельних рішень.

Наприклад, якщо модель є системою звичайних диференціальних рівнянь, то ці рівняння подаються не в алгебраїчній і не в лінійній формі. Моделі в інваріантній формі перед включенням у бібліотеку моделей конкретної програми повинні бути перетворені у форму, прийняту в даній

програмі. Серед цих форм варто виділити, в першу чергу, нормальну форму Коші і подання алгебро-диференціальних рівнянь в алгебраїчній і лінійній формах. В останньому випадку модель є алгоритмом для обчислення тих елементів матриці Якобі, що залежать від параметрів моделі.

3. *Макромоделі.* Доцільним є створення макромоделей для всіх мікросхем, що серійно випускаються, і для типових ВІС (великі інтегральні схеми) і НВІС (надвеликі інтегральні схеми).

Розрізняють макромоделі факторні, фазові, змішані.

- *Факторні* (логічні) макромоделі. Ці макромоделі призначені для використання в програмах аналізу логічних схем. Тому їхню основу складають логічні рівняння, що виражають залежності вихідних булевих змінних від вхідних булевих змінних. Але, на відміну від найпростіших моделей елементів функціонально-логічного рівня, у факторній макромоделі додатково включаються вирази, що пов'язують вихідні і зовнішні параметри (залежності потужностей розсіювання і затримок поширення сигналів від температури навколишнього середовища, напруг джерел живлення, параметрів навантаження і т.п.)
- *Фазові* (електричні) макромоделі. Дані макромоделі призначені для використання в програмах аналізу електронних схем на схемотехнічному рівні. Тому вони пов'язують між собою такі фазові змінні, як струми і напруги. Схемне зображення електричних макромоделей задається у вигляді електричних схем із двополюсниками типу опорів, ємностей, залежних джерел і т.п.
- *Змішані* макромоделі. Ці макромоделі призначені для аналізу цифрових і цифро-аналогових схем на стику функціонально-логічного і схемотехнічного проектування. У змішаній макромоделі елемент з боку входів і з боку виходів подається по-різному. Наприклад, на входах використовуються схемотехнічні зображення, типові для аналізу електронних схем, а на виходах – логічні подання, типові для дослідження функціональних схем цифрових пристроїв.

3.5.2 Приклад складання структурної моделі цифрової схеми

У структурній моделі елементний склад принципової схеми і вхідний вплив задаються одновимірними векторами. Міжелементні зв'язки, розташування зовнішніх входів і виходів задаються багатовимірною матрицею.

Для прикладу розглянемо принципову схему, представлену на рисунку 3.1.

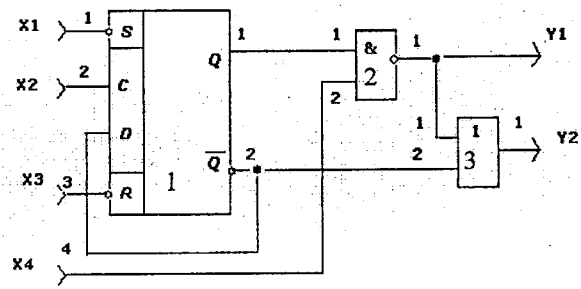


Рисунок 3.1 – Принципова схема для складання структурної моделі

Перша матриця – задає елементний склад схеми, що моделюється. Вона є одномірнм масивом, кількість елементів якого дорівнює кількості елементів у схемі для моделювання. Така ж кількість стовпців буде у всіх інших матрицях. У цьому масиві зазначається умовне позначення елементів або їхній код. Нумерація елементів схеми може бути довільною, але для збільшення швидкодії моделі необхідно дотримуватися правила: більш менший номер повинен мати елемент, що має більше зовнішніх входів. Для схеми, поданої, на рисунку 3.1, цей масив буде мати вигляд:

Тг	2І-НІ	2АБО-НІ

Друга матриця – задає значення виходів елементів схеми. Кількість рядків цієї матриці визначається максимальною кількістю виходів в елементах схеми. Для наведеної схеми кількість рядків буде два:

0	Z	Z
1	X	X

Z – стан виходу не визначено;

X – вихід фізично відсутній.

Третя матриця – задає розташування зовнішніх входів. Для наведеної схеми вона буде мати вигляд:

1	X	X
2	4	X
X	X	X
3	X	X

Наступні дві таблиці використовуються для задання внутрішньосхемних з'єднань. Кількість рядків у них задається максимальною кількістю входів в елементах схеми.

Четверта матриця – задає номери виходів елементів, з якими зв'язаний даний вхід елемента.

Вн	1	1
Вн	Вн	2
2	X	X
Вн	X	X

П'ята – указує номери елементів, виходи яких зазначені в попередній таблиці.

Вн	1	2
Вн	Вн	1
1	X	X
Вн	X	X

Шоста матриця – задає розташування зовнішніх виходів схеми. Кількість її рядків визначається максимальним значенням виходів елементів схеми.

X	1	2
X	X	X

Контрольні запитання

1. Склад та характеристики математичного забезпечення САПР.
2. Моделювання при проектуванні технічних засобів.
3. Класифікація математичних моделей.
4. Вибір типу моделі, визначення змісту, параметрів та характеристик моделей.
5. Принципи побудови математичної моделі цифрової схеми.
6. Формування математичних моделей принципових логічних схем.
7. Моделювання як основний апарат і засіб автоматизованого проектування ЕОМ.

4 ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ (ТЗаб) САПР

Розробка САПР – це комплекс взаємозалежних робіт зі створення математичного, програмного, технічного, інформаційного й іншого видів забезпечення систем, орієнтованих на автоматизоване проектування певного класу об'єктів (САПР машинобудування, літакобудування, БІС, ЕОМ і ін.).

У розробці і впровадженні САПР беруть участь великі колективи проектних і конструкторсько-технологічних організацій, зусилля яких координуються групою системних координаторів.

У цьому розділі розглянемо деякі специфічні аспекти розробки технічного забезпечення САПР.

4.1 Розробка ТЗаб САПР

До ТЗаб САПР висуваються вимоги можливості організації оперативної взаємодії проектувальників з ЕОМ, достатньої продуктивності обчислювальних засобів і необхідного об'єму оперативної пам'яті для розв'язання задач автоматизованого проектування за прийнятний час, можливості одночасної роботи декількох користувачів з ресурсами ТЗ, високої надійності, прийнятої вартості і т.п.

Дотримання перерахованих вимог можливе тільки шляхом організації ТЗаб САПР у вигляді спеціалізованої ієрархічної обчислювальної системи чи обчислювальної мережі з розвинутим периферійним обладнанням, які орієнтовані на введення, обробку і видачу текстової і графічної інформації.

Задача розробки ТЗаб САПР полягає в обґрунтуванні, розрахунку і виборі структури багаторівневого комплексу технічних засобів (КТЗ) САПР, орієнтованого на розв'язання задач автоматизованого проектування певного класу об'єктів. Побудова КТЗ може здійснюватися шляхом комплектування як стандартного обладнання (ЕОМ, канали, дисплеї, пристрої зовнішньої пам'яті і т.д.), так і спеціально розробленого для КТЗ (АРМ, графопобудовники і т.д.).

Створення багаторівневих КТЗ припускає наявність на вищому рівні однієї чи кількох ЕОМ великої продуктивності. Ці ЕОМ призначені для розв'язання складних задач проектування, що вимагають великих витрат машинного часу і пам'яті. На нижчих рівнях ієрархії можуть знаходитися ЕОМ середньої продуктивності, а також міні- і мікро-ЕОМ, що входять до складу АРМ (термінальні ЕОМ). Ці ЕОМ призначені для розв'язання порівняно нескладних задач проектування, для керування роботою комплекту периферійного обладнання і для організації обміну інформацією між різними рівнями КТЗ.

Для визначення структури КТЗ і параметрів компонентів, що входять до його складу, можуть служити обмеження:

- знизу – на число програм N, що входять до складу програмного забезпечення САПР, та на середній час T_n реакції КТЗ на задачу проектування, що надійшла;
- зверху – на обсяг оперативної пам'яті E_n для збереження програм проектування та на час T_n, необхідний процесору для розв'язання усередненої задачі в однопрограмному режимі, а також номенклатура периферійного обладнання КТЗ САПР.

Комплекси технічних засобів САПР створюються на базі засобів обчислювальної техніки загального призначення – супер-ЕОМ, «Ельбрус», міні- і мікро-ЕОМ різних типів.

Супер-ЕОМ – це сукупність технічних засобів і програмного забезпечення, на основі яких можна створювати обчислювальні системи різної конфігурації надвисокої продуктивності.

Концепції, закладені в супер-ЕОМ (програмна сумісність, універсальність, модульний принцип побудови технічних засобів і програмного забезпечення), дозволяють вдосконалювати усі компоненти системи. За допомогою набору команд супер-ЕОМ виконують операції з фіксованою і плаваючою комами, десяткові операції й операції з полями змінної довжини.

Система програмного забезпечення супер-ЕОМ складається з операційних систем, пакетів прикладних програм і програм технічного обслуговування. У старших моделях супер-ЕОМ використовується принцип конвеєрної обробки.

На основі супер-ЕОМ створюються багатомашинні і багатопроцесорні комплекси. У багатомашинній системі кожна ЕОМ керується власною ОС на відміну від багатопроцесорної, де операційна система загальна.

Багатопроцесорні обчислювальні комплекси призначені для розв'язання різних науково-технічних задач, у тому числі і задач автоматизованого проектування, з високими вимогами до продуктивності і стабільності обчислювальної системи.

Одна з головних особливостей таких комплексів – наявність в їх складі спеціалізованого процесора, що працює в режимі апаратної емуляції і дозволяє забезпечувати реалізацію великої бібліотеки прикладних програм.

Ці комплекси випускають декількох типів, структурні і логічні рішення яких однакові, а відмінність полягає в числі використовуваних центральних і спеціалізованих процесорів, у об'ємі ОЗП й у кількісному складі зовнішніх запам'ятовувальних пристроїв.

Основними особливостями структури центрального процесора є безадресна система команд, динамічний розподіл надоперативних регістрів, робота з полями змінної довжини, віртуальна пам'ять об'ємом

2^{32} слів, розподіл оперативної пам'яті сегментами змінної довжини, організація паралельних процесів і т.п.

Багатомашинні обчислювальні комплекси мають високі показники надійності і достовірності обробки інформації за рахунок модульного принципу побудови і наявності системи реконфігурації, яка при виникненні сигналу несправності від системи апаратного контролю автоматично виключає його зі складу комплексу і відновлює перерваний обчислювальний процес.

Сучасні персональні ЕОМ мають великі функціональні і технічні можливості. Оснащені проблемно-орієнтованим програмним забезпеченням і спеціальним периферійним обладнанням, вони стають найважливішими компонентами ТЗаб САПР.

Широке застосування персональні ЕОМ знаходять в організації автоматизованих робочих місць. Автоматизовані робочі місця є основним технічним засобом зв'язку оператора з КТЗ. Для досягнення найбільшої ефективності зв'язку до складу АРМ, зазвичай, включають дисплей з роздільною здатністю, світловим пером, а також велику кількість цифрових перетворювачів для конструювання графічних проектів.

4.2 Специфічні технічні засоби

4.2.1 Пристрої машинної графіки

Пристрої машинної графіки (ПМГ) підрозділяються на пасивні й активні. *Пасивні* застосовують тільки для виведення інформації, *активні* – для введення.

За методами реєстрації ПМГ поділяються на *точкові* і *векторні*. У точкових зображення формується з точок, у векторних – з найпростіших елементів – примітивів.

Основними ПМГ є креслярські автомати (КА) і кодувальники графічної інформації. За принципом дії розрізняють КА електромеханічні, електронні, оптико-механічні і рядково-реєстрові.

В *електромеханічних* КА електричні керувальні сигнали перетворюються в переміщення друкувального вузла. Такі КА забезпечують високу точність і якість зображення. Швидкість роботи КА цього класу обмежена інерційністю електромеханічних вузлів.

В *електронних* КА виконавчим органом служить електронно-променева трубка. Прикладом електронного КА може бути графічний дисплей.

Для реєстрації зображення в *оптико-механічних* КА використовується модульований за яскравістю світловий промінь, що сканує по світлочутливому папері. Подібні КА застосовуються рідко через складність обробки паперу й обмежені розміри зображення.

У *рядково-реєстрових* КА зображення також формується на спеціальному (електромеханічному, електротермічному) папері, що дуже

незручно. Крім того, зображення виходить невисокої якості. Однак рядково-реєстрові КА мають високу швидкість відтворення інформації.

Розглянемо детальніше розповсюджені електромеханічні КА. Залежно від точності і швидкодії їх поділяють на координатографи і графопобудовники (ГП).

Координатографи працюють з меншою, чим ГП, швидкістю, але з вищою точністю. Швидкість роботи може складати кілька десятків міліметрів у секунду, а погрішність – не більше 0,05 мм. У координатографах використовуються такі способи нанесення малюнка: вирізання, гравірування, експонування, наколювання точок голкою і т. п. Як носії зображення застосовують фотопластини, фотоплівки, пластини із шаром лаку і із шаром емалі. Координатографи використовують, наприклад, для одержання фотошаблонів і друкованих плат.

Матеріал, на який координатограф наносить зображення, жорстко фіксується на нерухомому столі. Реєструвальний пристрій (електромеханічного принципу дії) може переміщатися в одному з двох ортогональних напрямків. Для реєстрації рисунка координатографи мають спеціальні інструменти: фотоголовки, креслярські головки, фрезерувальні головки, головки, що ріжуть і т. п. Цими інструментами виконується рисунок на носіях зображення. Фотоголовка за допомогою поворотного диска з отворами формує світловий промінь заданої товщини для реєстрації контурного зображення, може створювати світлову пляму для динамічного зображення топологічного малюнка, вдруковувати фрагменти потрібної форми (наприклад, прямокутної). Креслярська головка комплектується кульковими і креслярськими ручками, олівцями чи стержнями. Спосіб нанесення малюнка в цьому випадку не відрізняється від традиційного.

Не менш важливою характеристикою координатографа є розмір робочого поля.

Графопобудовники (плотери) працюють з більшою швидкістю (до 1 м/с), але з меншою точністю (більше 0,02 мм). Як реєструвальний пристрій тут застосовуються різнокольорові пера, фломастери, стержні, ручки, олівці. Для одержання ліній різної товщини використовують до шести пер.

Зображення креслення виходить на аркушах папера, що закріплені на планшеті або на рулонному папері. У зв'язку з цим, розрізняють планшетні, рулонні, рулонно-планшетні і барабанні ГП.

У *планшетних* ГП папір жорстко кріпиться до столі-планшеті механічним чи вакуумним способом, а реєструвальний пристрій, переміщається по двох координатах. Планшетні ГП використовують для одержання складних креслень і схем при безперервному контролі результатів роботи.

У *рулонних* ГП папір із крайовою перфорацією переміщається по одній осі, а реєструвальний пристрій – по іншій. Рулонні ГП більш

компактні, але мають менший розмір робочого поля й поступаються планшетним за якістю підготовлюваних креслень. Погіршення якості пояснюється погрішностями "прогону" паперу і якості самого паперу.

Рулонно-планшетні ГП використовують принципи рулонної і планшетної систем. Для поліпшення якості креслень застосовується спеціальний папір високої якості. Прикладом таких ГП є пристрої "КалКомп" (США), за допомогою яких можна виготовляти багатобарвні креслення з високою точністю (розподільна здатність – до 400 точок на дюйм) і великою продуктивністю (до 940 мм/с). Зображення креслення видається на папір шириною до 1094 мм. Для керування операціями ГП розроблене спеціальне програмне забезпечення на ЕОМ IBM PC XT, AT.

Найбільшу швидкість мають *барабанні* ГП. При виведенні графічної інформації на такому плотері папір кріпиться липкою стрічкою до поверхні металевого барабана з довжиною кола 1200 мм, що обертається в обох напрямках. Прикладом барабанного ГП може служити пристрій Benson5342 (Франція), у якого розмір робочого поля досягає 840×1200 мм, а швидкість роботи — 1130 мм/с.

КА за способом зв'язку з ЕОМ бувають *автономними,* безпосередньо підключеними до ЕОМ чи *універсальними.* Виконання складних креслень займає декілька годин.

Переміщення реєструвального пристрою у КА здійснюється за допомогою електродвигунів постійного струму чи крокових електродвигунів.

Для роботи КА повинна бути підготовлена (засобами САПР) керувальна інформація, що включає:

- номер використовуваного пера;
- тип лінії (суцільна, пунктирна, штрихова, штрихпунктирна);
- вид інтерполяції;
- координати точок;
- ознака включення пера (підняте чи опущене);
- форму лінії (відрізок прямої, дуга, символ).

Для креслення стандартних символів (букв) застосовуються спеціальні генератори знаків.

Вище були розглянуті пасивні ПМГ, що широко використовуються в САПР.

Для введення графічної інформації в ЕОМ використовуються спеціальні ПМГ – пристрої введення графічної інформації (ПВГІ). Інформація, що вводиться, спочатку зчитується, а потім кодується. Тому ПВГІ називають *кодувальниками.* Під *зчитуванням* розуміють розпізнавання графічного елемента (точка, лінія, елементарне зображення – примітив) і визначення його координат у прийнятій системі. *Кодуванням* називається перетворення зчитаної інформації в цифровий код за встановленими правилами.

Залежно від реалізації операції зчитування розрізняють автоматичні і напівавтоматичні кодувальники. *Автоматичні кодувальники* використовують відслідковувальний і розгортаючий (сканувальний) методи перетворення. У першому випадку робочий пристрій відслідковує границю заданої кривої, переміщаючись з постійною швидкістю по осі абсцис. Крива, що вводиться, задається сукупністю числових значень відхилень робочого пристрою по осі ординат. В другому випадку здійснюється сканування зображення робочим пристроєм з деяким кроком по осі абсцис. При цьому фіксуються ординати точок перетинання скануючим променем заданої кривої. Автоматичні кодувальники застосовуються для підготовки простих малюнків.

Для введення складних малюнків і креслень застосовуються *напівавтоматичні кодувальники.* Операцію зчитування в цьому випадку виконує людина за допомогою спеціального щупа. Зчитана інформація приймається і кодується електронним блоком. Закодовану інформацію можна записати на проміжний носій (магнітну стрічку) чи відразу передати в ЕОМ через блок сполучення з каналом введення-виведення.

Інформація, що вводиться (малюнок), міститься на робочому полі – планшеті пристрою, що обладнаний алфавітно-цифровою функціональною клавіатурою.

4.2.2 Автоматизовані робочі місця проектувальників

Комплекси технічних засобів САПР першого покоління – це універсальна ЕОМ з мінімальним набором периферійних пристроїв. За допомогою таких засобів можна були розв'язувати лише окремі задачі на деяких етапах проектування.

У сучасних умовах для підвищення ефективності САПР використовуються спеціалізовані проблемно-орієнтовані обчислювальні системи із великим об'ємом оперативної пам'яті, зручними і швидкими засобами взаємодії проектувальника з ЕОМ для одержання проектної документації.

Для одержання необхідних характеристик розроблюваної САПР на основі розглянутих вище принципів визначається конфігурація КТЗ САПР – розробляється ТЗ конкретної САПР. При визначенні конфігурації КТЗ САПР враховують:

- число рівнів в КТЗ;
- задачі, розв'язувані на кожному рівні;
- територіальне розміщення технічних засобів;
- характер зв'язків між окремими підрозділами проектної організації й обсяг інформації, якою ці підрозділи повинні обмінюватися;
- технічну стратегію розробки, експлуатації і перспектив розвитку КТЗ САПР.

КТЗ САПР може містити до чотирьох рівнів. Найчастіше в даний час

застосовуються однорівневі КТЗ, прикладом яких може служити універсальна ЕОМ, оснащена периферійними пристроями і засобами організації діалогу. Для деяких САПР такий (стандартний) комплект цілком прийнятний. У той самий час застосування базовою машиною ЕОМ, що працює в мультипрограмному режимі, через відсутність зручних засобів зв'язку проектувальника із системою і засобів введення-виведення графічної інформації для других САПР виявилося неефективним. Тому були розпочаті спроби створення проблемно-орієнтованого обчислювального комплексу, що одержав назву *автоматизованого робочого місця проектувальника* (АРМ), структуру якого наведено на рисунку 4.1.

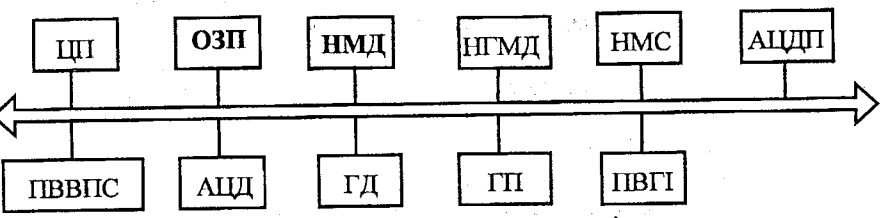


Рисунок 4.1 – Структура АРМ першого покоління:

ЦП – центральний процесор;
ОЗП – оперативно-запам'ятовувальний пристрій;
НМД – накопичувач на магнітних дисках;
НМС – накопичувач на магнітних стрічках;
АЦДП – алфавітно-цифровий друкувальний пристрій;
ПВВПС – пристрій введення-виведення на перфострічку;
АЦД – алфавітно-цифровий дисплей;
ГД – графічний дисплей;
ГП – графопобудовник;
ПВГІ – пристрій введення графічної інформації.

АРМ – це набір *програмно-керованих пристроїв*, що об'єднані продуктивною ЕОМ з розвинутою *системою загального і спеціального програмного забезпечення*. АРМ орієнтовані на розв'язання задач з підготовки, перетворення і редагування текстової і графічної інформації, одержання документації на машинних носіях, розв'язання схемотехнічних і конструкторських задач в автономному режимі.

З їхньою допомогою також вибираються задачі з організації обміну із САПР, тому що АРМ може працювати автономно, а також у режимі безпосереднього зв'язку з центральною ЕОМ.

АРМ першого покоління, створені на базі міні-ЕОМ ("Наірі-4", М 4000, СМ 3, СМ 4) з досить широким набором периферійних пристроїв,

мали структуру, що показана на рисунку 4.1. В автономному режимі вони використовувалися для розв'язання окремих задач, що не вимагають великого об'єму пам'яті (напівавтоматизоване проектування друкованих плат з випуском комплекту текстової і графічної документації і керувальних перфострічок, проектування фотошаблонів топології мікросхем, підготовка керувальних перфострічок для верстатів із ЧПУ і т.п.).

У режимі безпосереднього зв'язку з центральною ЕОМ на АРМ розв'язувалися такі задачі:

- введення і редагування великих масивів вхідної інформації;
- керування режимами роботи САПР;
- відображення і редагування результатів проектування при розв'язанні складних задач (моделювання, оптимізація електронних схем цифрової й аналогової апаратури);
- компонування і трасування плат друкованого монтажу і мікрозборок;
- синтез механічних конструкцій;
- створення і поповнення банку даних.

У такому режимі основні процедури проектування виконувалися на потужній ЕОМ, а АРМ служив для організації діалогового режиму роботи САПР.

З вітчизняних АРМ першого покоління найбільш відомі АРМ-Р (орієнтований на проектування радіоелектронної апаратури) і АРМ-М (для проектування машинобудівних конструкцій).

АРМ-Р випускався різних конфігурацій. Мінімальний базовий комплект АРМ-Р-01 включав: ЕОМ М 4000 із ЦП і пультом керування, ОЗП обсягом 8 Кбайт, блоком загальної 16-розрядної шини, системою електроживлення, пристроями введення-виведення на перфострічку і друкарську машинку; пристрій розширення пам'яті; НМД ("ІЗОТ-1370"); АЦД (VT340).

У комплекті АРМ-Р-02 до наявного обладнання додано графічний дисплей ЭПГ-400. Цих засобів виявилося досить для автоматизованого розміщення елементів на платі, редагування текстової і графічної інформації, організації діалогу із САПР на ЕОМ ЄС.

У комплекті АРМ-Р-03 до АРМ-Р-01 додано пристрій мозаїчного друку DZM-180. Завдяки цьому комплект можна було застосовувати для налагодження.

АРМ-Р-04, додатково до АРМ-Р-01, включає графопобудовник АП-7252.

До складу АРМ-Р-05, крім пристроїв АРМ-Р-01, додані DZM-180, графічний дисплей ЭЛГ-400, чотири графопобудовники, кодувальник ЭМ-709.

АРМ-Р-06 містить три кодувальники ЭМ-709.

Все устаткування, що входить до складу АРМ-Р, об'єднано в систему за допомогою загальної шини (ЗШ), що забезпечує єдиний алгоритм зв'язку між різними компонентами і структурно дозволяє підключати значне число пристроїв.

Адреси, дані і керувальна інформація передаються по 56 лініях шини, що двонаправлені – їх можна використовувати як вхідні, так і вихідні.

Зв'язок по шині організований за принципом "запит-відповідь". На кожен сигнал керування керувальний пристрій (задавальник) одержує відповідь від виконавця (керованого пристрою). Така організація обміну дозволяє не накладати значних обмежень на фізичну довжину загальної шини і тривалість сигналів обміну.

АРМ-М має чотири конфігурації.

АРМ першого покоління розраховані на одного проектувальника, а в проектуванні, зазвичай, бере участь цілий колектив. Недоліками таких комплексів є висока вартість і складність розробки програмного забезпечення.

Ефективнішими виявилися АРМ другого покоління, що відбили тенденції розвитку обчислювальної техніки.

Багатотермінальні АРМ створювалися на базі ЕОМ СМ 4 , СМ 1420, "Електроніка 79". Вони розраховані на обслуговування групи проектувальників. Склад пристроїв комплексу, що одержав назву *інженерної робочої станції* (ІРС), показаний на рисунку 4.2.

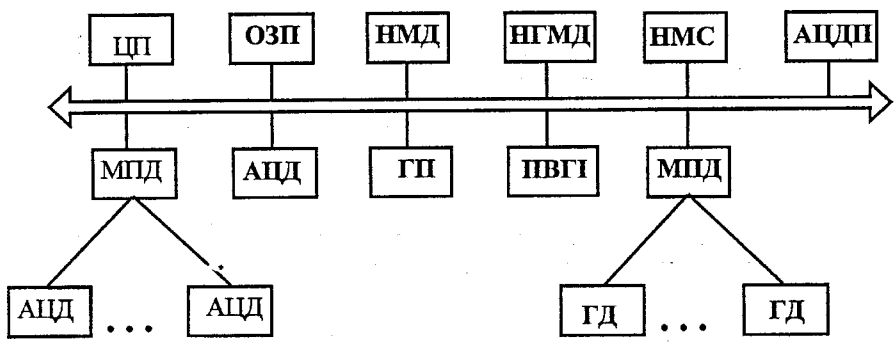


Рисунок 4.2 – Інженерна робоча станція

Використання ПЕОМ істотно знижує вартість АРМ другого покоління. Вони мають обмежений набір периферійних пристроїв, що орієнтовані на одного проектувальника. Такі комплекси називають також *робочим місцем проектувальника* (РМП). Прикладом персонального АРМ може служить АРМ-П, що має у своєму складі ПЕОМ ЄС 1841, графічний

дисплей (ГД), принтер і ГП.

Таким чином, розглянуто два варіанти однорівневих КТЗ САПР: ЕОМ і АРМ.

Відомі чотири варіанти дворівневих КТЗ САПР:

- ЦОК-АРМ;
- ЦОК-ТК;
- АРМ-ТК;
- АРМ-АРМ.

На рисунку 4.3 показані варіанти дворівневого КТЗ.

Найчастіше застосовують ієрархічну структуру з ЦОК на верхньому рівні. Нижній рівень утворюють ІРС, РМП чи технологічне устаткування. ЦОК керує роботою дворівневого КТЗ. Труднощі побудови такого варіанта КТЗ пов'язані з необхідністю програмної сумісності ЕОМ різних рівнів. Простішим варіантом є АРМ-АРМ, тому що в цьому випадку працюють програмно-спільні ЕОМ.

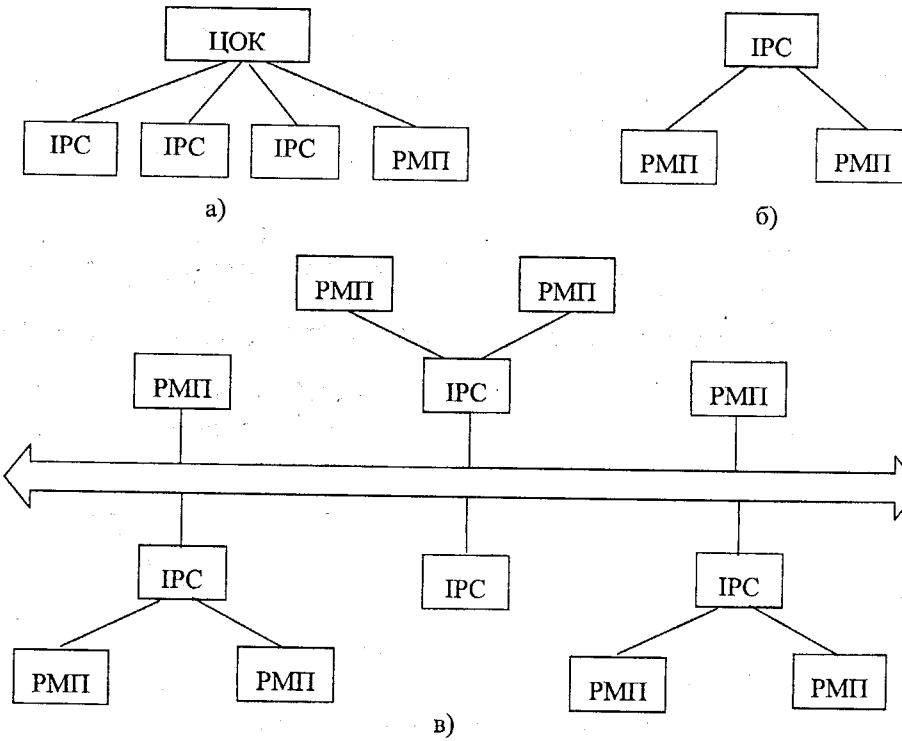


Рисунок 4.3 – Варіанти дворівневого КТЗ

а) – ЦОК – АРМ; б) – АРМ-АРМ; в) – АРМ-АРМ на базі

локальної обчислювальної мережі

Прикладом вітчизняного дворівневого КТЗ САПР є система 15УТ-1-060 ("КУЛОН-3"). До її складу входять: мікро-ЕОМ "Електроніка 79", ОЗП на 512 Кбайт, НМД, НМС, АЦДП, 16 АЦД, а також 8 РМП. Кожне РМП складається з мікро-ЕОМ "Електроніка 60М", ОЗП на 56 Кбайт, НМД ("ІЗОТ-1370"), НГМД, ПВГІ, ГП, АЦД і ГД (ЭМ-739А). "КУЛОН-3" орієнтований на автоматизоване проектування виробів електронної техніки, радіоелектронної апаратури і виробів машинобудування. Однак побудова САПР на його основі вимагає розробки відповідного програмного забезпечення, що є досить важкою задачею.

Закінчуючи розгляд багаторівневих КТС САПР, відзначимо наявність двох варіантів трирівневого КТС (ЦОК-АРМ-АРМ, ЦОК-АРМ-ТК) і варіанта чотирирівневого комплексу (ЦОК-АРМ-АРМ-ТК).

4.3 Режими роботи апаратури в КТЗ САПР

Комплекси технічних засобів САПР і автоматизовані робочі місця є системами колективного користування. Тому ЕОМ, що входять до складу ТЗ САПР, і їхні операційні системи повинні допускати одночасне розв'язання декількох задач від різних користувачів, тобто повинні працювати в режимі мультипрограмування.

За характером обміну інформацією між користувачем і ЕОМ розрізняють пакетний і діалоговий режими роботи. У *пакетному* режимі розв'язуються задачі, для яких можлива і доцільна повна формалізація і для розв'язання яких не потрібно оперативного втручання людини.

У *діалоговому* (інтерактивному) режимі розв'язуються задачі, для яких відсутні чи є неефективними формальні правила прийняття рішень і виконуються умови переваги діалогового режиму (невеликий час реакції системи на запит користувача і малий обсяг інформації, що вводиться користувачем при звертанні до ЕОМ). Характерною рисою КТЗ САПР є наявність діалогового режиму.

Мультипрограмний діалоговий режим роботи КТЗ називається режимом *розподілу часу*. Цей режим необхідний у САПР для організації одночасної роботи кількох користувачів САПР на загальних технічних засобах.

Окремі пристрої введення/виведення можуть працювати в режимі *on-line* (на лінії), де здійснюється безпосередній електричний зв'язок зовнішнього пристрою з термінальної ЕОМ, і в режимі *off-line* (поза лінією), де пристрій працює автономно.

4.4 Підходи до вибору структури КТЗ САПР

При проектуванні КТЗ САПР широке застосування знаходять методи оптимізації і моделювання.

Методи синтезу КТЗ САПР призначені для вибору оптимальної структури системи і її компонентів і базуються, в основному, на методах дослідження операцій – методах математичного програмування, теорії графів і мереж, теорії масового обслуговування та ін.

Методи аналізу КТЗ призначені для оцінки обчислювальної потужності комплексу і необхідного об'єму оперативної і зовнішньої пам'яті обчислювальних засобів і базуються на застосуванні методів імітаційного й аналітичного моделювання. Методи імітаційного моделювання дозволяють врахувати велике число параметрів і досягти високого ступеня адекватності при відповідному ускладненні моделі проектованого об'єкта. Однак процес побудови імітаційних моделей є досить трудомістким і вимагає як первинні методи оцінки структур ТЗ САПР використання аналітичних методів, які використовують для побудови моделей синтезу оптимальних структур.

Проектування КТЗ САПР починається з аналізу технічних вимог до системи проектування, підсумків передпроектного обстеження об'єктів проектування і вихідних даних. Як правило, попереднє комплектування структури технічного забезпечення САПР здійснюється за допомогою довідкових матеріалів.

Відповідно до номенклатури випуску обладнання визначають мінімальний склад технічних засобів, необхідний для реалізації заданого класу задач автоматизованого проектування, що дає можливість прийняти в розрахунках конкретні значення параметрів застосовуваних технічних засобів.

При розв'язанні задачі комплектування технічних засобів необхідно визначити типи застосованих ЕОМ, їхнє число й ієрархію для забезпечення розв'язання заданого набору задач у прийнятний для розробників час. При цьому варто враховувати, що частина задач може бути розв'язана в пакетному режимі, де на терміни розв'язання кожної задачі накладаються планові чи директивні обмеження (наприклад, кінець зміни), а основна маса задач автоматизованого проектування і конструювання повинна розв'язуватися в інтерактивному режимі і результат розв'язування кожної задачі повинен виходити у певному часовому інтервалі після подачі запиту на її розв'язання.

Створення і широке впровадження 64-розрядних мікро-ЕОМ і персональних комп'ютерів, збільшення об'єму їхньої оперативної пам'яті, розширення номенклатури і об'єму накопичувачів на магнітних дисках, поява ефективніших спеціалізованих операційних систем сприяють широкому їхньому використанню в складі КТЗ САПР. Однак при проектуванні ТЗ САПР необхідні детальні розрахунки системних характеристик застосування мікро-ЕОМ у складі КТЗ для того, щоб забезпечити досить ефективне їхнє використання. Створювані КТЗ САПР повинні забезпечувати ефективне використання пристроїв збору і передачі

інформації, гарантовані характеристики надійності КТЗ, можливість гнучкої зміни структури, номенклатури і кількості технічних засобів, що забезпечують поетапне запровадження в дію компонентів КТЗ і його модернізацію.

Розробка систем автоматизованого проектування є надзвичайно трудомістким процесом, у якому задіяні великі колективи фахівців. Вартість сучасних САПР досить висока, тому технічне, програмне й інформаційне забезпечення САПР у даний час складає значну частку основних виробничих фондів підприємств і організацій, що використовують САПР. Фондовіддача цієї складової залежить від того чи має користувач усі компоненти САПР, необхідні для організації автоматизованого проектування заданої номенклатури виробів, чи задовольняють користувача склад і якість КТЗ, програмного й інформаційного забезпечення САПР.

Виходячи з особливостей обчислювального процесу до САПР висуваються такі вимоги:

1. Наявність діалогового режиму, що випливає з об'єму проектування як процесу прийняття складних рішень в умовах неоднозначності планів обчислень. Конфліктна частина процесу проектування відображається інтерактивним терміналом, а процес прийняття рішень здійснюється проектувальником.

2. Колективний доступ користувачів до ресурсів системи. Доступ до ресурсів САПР означає можливість виклику будь-якого програмного модуля, звертання до бази даних, відображення результатів проектування на екрані дисплея, на принтері чи на графопобудовниках у вигляді конструкторської документації.

3. Робота програм з базами даних різної логічної структури і складності шляхом забезпечення незалежності програм від способу організації даних.

4. Можливість автономного налагодження програмного забезпечення САПР.

5. Можливість подальшого розвитку системи шляхом розвитку програмного, інформаційного і технічного забезпечення.

4.5 Оцінювання якості ТЗаб САПР

При наявності необхідного набору КТЗ, ПЗ і ІЗ визначальним стає якість САПР.

Під *якістю* розуміється сукупність властивостей виробу, що обумовлюють його придатність задовольняти визначені потреби відповідно до його призначення.

Кількісні характеристики однієї чи декількох властивостей називають *показниками якості.* Поліпшення показників якості проектованих САПР сприяє проведенню бездефектного

проектування технічних об'єктів і, як наслідок, – поліпшення їхньої якості.

Відповідно до державних стандартів оцінки якості виробів розглянемо такі групи показників для оцінки САПР: призначення, функціональні, експлуатаційні, технологічні, економічні і стандартизації.

1. *Показники призначення.* До цих показників відносяться галузь застосування САПР, розмірність розв'язуваних задач проектування і необхідних технічних засобів для реалізації САПР.

2. *Функціональні показники.* Показники цієї групи характеризують властивості САПР, що виявляються безпосередньо в процесі функціонування; точність, надійність, стійкість до зовнішніх впливів на систему.

Точність проектування залежить від багатьох факторів:

- від ступеня адекватності математичної моделі проектування і проектованого об'єкта;
- від похибки математичних методів, використовуваних при розв'язанні задач оптимізації;
- від похибки округлення величин і т.д.

Надійність САПР – це властивість системи не втрачати працездатність при наявності відмов і збоїв у КТЗ САПР (апаратурна надійність) і при перетворенні певного набору вихідних даних через використання при цьому програм, що містять помилки (програмна надійність).

Стійкість САПР до спотворювальних впливів – це здатність системи виконувати своє функціональне призначення при наявності зовнішніх і внутрішніх спотворювальних впливів. Джерелами зовнішніх впливів можуть бути мережі електроживлення, непристосовані для експлуатації обчислювальної техніки приміщення, помилки в підготовці даних і т.п. Джерелами внутрішніх впливів є збої і відмови в самому КТЗ САПР, що можуть призвести до перекручування кодів програми, результатів проектування і т.п. Кількісною оцінкою показника стійкості може служити такий критерій, як область стійкого функціонування. Задача визначення області стійкого функціонування САПР аналогічна задачі визначення допусків і технічних вимог.

Час реакції системи T_c характеризується часовим інтервалом між моментом надходження в КТЗ завдання на проектування і моментом видачі відповідної документації. Величина T_c є випадковою і залежить від характеристик використовуваних обчислювальних засобів, периферійного обладнання і трансляторів, структури програмного й інформаційного забезпечення, а також від структури алгоритмів проектування і розмірності розв'язуваних задач.

3. *Експлуатаційні показники.* Ці показники характеризують пристосованість САПР до експлуатації і містять у собі простоту освоєння і підготовки до роботи, діагностування, кількість і необхідну кваліфікацію обслуги, можливість внесення виправлень і доробок при зміні параметрів проектованих об'єктів і вимог до проектування.

4. *Технологічні показники.* До технологічних відносяться показники розподілу витрат ресурсів і часу на введення САПР в експлуатацію, підтримку її роботоздатності і супровід у специфічних умовах конкретного користувача.

5. *Економічні показники.* Ці показники характеризують витрати на розробки придбання усіх видів забезпечення САПР, створення і тестування дослідного зразка системи, тиражування, освоєння, обслуговування і супровід САПР.

6. *Рівень стандартизації.* Перевірка цього рівня здійснюється шляхом проведення нормоконтролю всієї документації на відповідність вимогам ДСТУ і ЄСПД.

Важливою характеристикою САПР є ступінь її інформаційного зв'язку з навколишнім середовищем. Систему називають *статичною,* якщо в процесі проектування не потрібно інформації про стан зовнішнього середовища в даний момент часу, і *динамічною,* якщо при своєму функціонуванні система постійно використовує інформацію про стан зовнішнього середовища з джерел, що знаходяться поза системою проектування.

Аналізуючи безліч наведених показників, неважко визначити, що одна група показників є вимірюваною, а інша – не вимірюваною. З погляду корисності обидві ці групи можуть мати однакове значення, тому що певним чином містять інформацію про властивості САПР. До вимірюваних характеристик відносяться всі функціональні, економічні і більшість експлуатаційних показників, а до не вимірюваних – галузь застосування, необхідна кваліфікація обслуги і т.п. При оцінюванні не вимірюваних показників доцільно використовувати експертні методи.

Системи автоматизованого проектування – це сукупність технічних, програмних, інформаційних і інших засобів, що знаходяться між собою в тісному взаємозв'язку, тому на показники якості САПР істотний вплив мають показники якості складових компонентів системи .

Розглянемо деякі показники якості технічного, програмного і лінгвістичного забезпечення САПР.

Показники якості можна розділити на дві групи:

- *системні показники,* які відбивають найзагальніші властивості технічних засобів і характеризують вимоги користувачів апаратури до ефективності її експлуатації (вартість, продуктивність обчислювальних засобів, об'єм пам'яті, число користувачів, число

алфавітно-цифрових і графічних терміналів, завантаженість окремих пристроїв і ін.);

- *користувацькі показники*, що відбивають специфічні вимоги конкретних типів систем автоматизованого проектування (режими роботи апаратури, застосовувані мови, швидкість і точність переробки інформації, організація діалогового режиму й ін.).

У зв'язку з розширенням сфери застосування засобів обчислювальної техніки і збільшенням обсягів обчислювальних робіт ефективна організація використання ЕОМ набула державного значення. Тому система показників оцінки якості КТЗ САПР повинна враховувати ступінь завантаження ЕОМ і інших технічних засобів, що входять до складу КТЗ.

4.5.1 Ефективність операційної системи

Ефективність функціонування обчислювальних засобів ТЗ САПР істотно залежить від ефективності роботи операційної системи. В обчислювальних системах з наявністю засобів генерації ОС у користувачів є можливість вибору конкретного варіанта структури ОС з урахуванням наявної конфігурації технічних засобів, класів розв'язуваних задач і необхідних режимів використання (пакетна обробка, режим телеобробки і т.д.).

При роботі ОС споживає значну кількість ресурсів ВР, що призводить до перекручування реального обчислювального процесу. Якість обчислювального процесу, здійснюваного ОС, оцінюється, насамперед, ступенем корисного використання ресурсів і витратою ресурсів на організацію обчислювального процесу. Ефективність використання ЦП C_n може бути поліпшена або шляхом скорочення часу перебування ЦП у стані очікування (за рахунок перерозподілу бібліотек між НМД і файлів на окремих НМД) або шляхом скорочення витрат на роботу програм ОС (перекомпіляцією ядра ОС, корекцією розміщення модулів ОС, корекцією використання системних керувальних програм).

Якість генерації ОС оцінюють вимірювальними моніторами, що фіксують зміну поточних станів ЦП і відповідних їм параметрів.

4.5.2 Оцінка якості програмного забезпечення САПР

Розробка програмного забезпечення САПР є одним з основних і відповідальних етапів створення САПР. При постійному розширенні ПЗ САПР витрати на ПЗ безупинно ростуть і, як правило, значно перевищують витрати на ТЗ САПР. Розробка ПЗ на сьогодні поставлена на індустріальну основу. Кінцевим продуктом розробки ПЗ є *програмний виріб* (ПВ), тобто програми, записані на носіях даних і минулі стадії промислової розробки, а також комплект документації, що супроводжує ПВ.

Ефективність функціонування САПР багато в чому визначається якістю програмних виробів. Низька якість ПВ завдає великих матеріальних

збитків, викликає недовіру до ПВ, відволікає фахівців користувачів від основної роботи. Вартість локалізації й усунення помилок в експлуатованому ПВ обходиться в десятки разів дорожче, ніж усунення помилок на стадії проектування.

Оцінка якості ПВ – задача багатофакторна і важко формалізується. Визначальним фактором при оцінці якості програми є різноманіття інтересів користувача. З цієї причини неможливо запропонувати якунебудь єдину універсальну міру якості ПВ, тут потрібно безліч характеристик, що охоплюють цілий спектр бажаних властивостей. У літературі зустрічаються різні набори показників якості ПВ. Розглянемо основні з них.

1. *Документованість* – характеристика програмного забезпечення, що поєднує властивості зрозумілості, і завершеності. ПЗ має властивість зрозумілості, якщо воно дозволяє користувачу зрозуміти призначення програмних засобів.

Програмний виріб має властивість прийнятності, якщо його документація не містить надлишкової інформації і не допускає багатозначної інтерпретації термінів і символів, і завершеності, якщо в ньому наявні усі необхідні компоненти, кожний з який розроблений всебічно.

2. *Ефективність* – властивість ПВ виконувати необхідні функції без зайвих витрат ресурсів. Як оцінку ефективності можна прийняти характеристику програми, значення якої прямо пропорційно швидкодії і обернено пропорційно об'єму використовуваних ресурсів. Термін «ресурси» треба розуміти в самому широкому змісті: це можуть бути оперативна і зовнішня пам'ять, пристрої графічного введення, графопобудовники, пропускна здатність каналу і т.д.

3. *Надійність* – властивість ПВ стійко виконувати необхідні функції. Надійність, насамперед, має на увазі відсутність помилки в програмі, але так як вони неминучі, то програма повинна бути побудована так, щоб усі помилки можна було просто виправити, а це можливо при наявності якісної програмної документації.

4. *Простота використання* – властивість ПВ нормально функціонувати в чітко визначеній галузі застосування даного ПВ.

5. *Зручність експлуатації* — властивість ПВ адаптуватися відповідно до нових вимог. Зручність експлуатації припускає зрозумілість, оціпювання і простоту внесення змін.

6. *Мобільність* – властивість ПВ, що полягає в його пристосованості до переносу на ЕОМ іншого типу, а також до зміни операційної системи.

7. *Сумісність* – властивість ПВ, що полягає в тому, що ПВ і програми, які входять до його складу, повинні виконуватися в технічному, інформаційному і програмному середовищі іншого типу, ніж те, для якого вони безпосередньо призначені.

8. *Випробовуваність* – властивість ПВ, що полягає в наявності можливості досить просто оцінювати правильність функціонування програми в умовах конкретного середовища.

9. *Вартість програми* є функцією всіх її характеристик.

Поліпшення кожної з характеристик впливає на вартість ПВ, тому завжди повинен бути знайдений розумний компроміс між ступенем поліпшення характеристики ПВ і збільшенням його вартості.

Безпосередня оцінка виділених характеристик дуже складна. Для одержання якісного програмного забезпечення необхідно приймати спеціальні міри, спрямовані на гарантування заданих характеристик якості. Аналіз якісних характеристик повинен проводитися на кожному етапі процесу розробки і створення ПВ.

4.5.3 Оцінка якості лінгвістичного забезпечення САПР

Сучасні ЕОМ у своєму програмному забезпеченні мають компілятори з різних мов програмування. Вибір мови програмування, найбільш придатної для розв'язуваних задач, значною мірою впливає і на якість ПЗ САПР. Тому одним з основних факторів при виборі мови програмування є відповідність мови програмування типу розв'язуваної задачі. Невідповідність мови розв'язуваній задачі створює труднощі в написанні і налагодженні програми.

Мова програмування повинна мати ефективний засіб для встановлення угод і регламентів, що забезпечують швидке і надійне ущільнення окремо створюваних частин загальної програми.

Критеріями якості мови програмування і транслятора є простота, надійність, швидкість трансляції, ефективність об'єктного коду, зручність при читанні і блочність структури. Розглянемо ці якості докладніше.

Простота. Засоби мови повинні бути простими для розуміння і виражені в простій і інтуїтивно зрозумілій для програмування формі, що є природною основою для спілкування програміста з машиною. Простоті мови сприяють однаковість у символіці й організації, цілісність основних концепцій.

Надійність. Під надійністю мови розуміється її властивість, що характеризується імовірністю безпомилкового написання і трансляції програми заданого розміру.

Швидкість трансляції. Швидкість трансляції – істотний показник того, що мови програмування не виключають етап налагодження в процесі розробок програми і досить часто програма багато разів транслюється в процесі налагодження, а виконується тільки один раз. Одним зі шляхів збільшення швидкості трансляції є використання незалежної трансляції, тобто компілятор може транслювати програму частинами.

Ефективність об'єктного коду. Одержання ефективних машинних кодів можливо в процесі трансляції тільки з мов, при розробці яких передбачалася оптимізація програм компілятором (мови ФОРТРАН, АДА).

Наявність гарних компіляторів дозволяє одержувати програми прийнятних розмірів і ефективності. Машинно-незалежні оптимізатори повинні перетворити неефективні програми в ефективніші, цілком еквівалентні і записані на тій самій вихідній мові.

Зручність при читанні. Від того, наскільки вдалі коментарі і як зручно вони розташовані в програмі, істотно залежить швидкість розуміння програми. Програми пишуться програмістами, читаються ними ж чи їхніми колегами ще до введення в ЕОМ, тому якості коментарів при розробці мови повинно надаватися великої уваги.

Блочність структури. Блокову структуру мають програми, написані мовами АЛГОЛ, ПЛ/1, АДА. У цих мовах кожна програма, чи підпрограма, організовується у вигляді послідовності вкладених один в одного блоків, обмежених спеціальними покажчиками (наприклад, в АЛГОЛ – словами begin і end).

Блокова структура забезпечує гнучку систему керування даними і дозволяє одержати високоефективні виконувані об'єктні коди.

Ясність структури програми дає програмісту і користувачу великі переваги. Програму, структура якої відбиває структуру реалізованого в ній алгоритму, легко проектувати, кодувати, налагоджувати, розуміти і супроводжувати.

4.6 Персональні ЕОМ в САПР

ПЕОМ – це невелика за розмірами і вартістю універсальна мікро-ЕОМ, призначена для індивідуального користувача. З конструктивної точки зору в ПЕОМ можна виділити п'ять основних пристроїв: системний блок, клавіатуру, дисплей, накопичувачі на жорсткому і гнучкому магнітному дисках, принтер.

До складу ПЕОМ можуть входити й інші пристрої, іноді частина пристроїв, з перерахованих вище, компонується у вигляді єдиного блоку.

Системний блок містить всю електронну начинку ПЕОМ. Його головні компоненти – ЦП і ОЗП. Основою ЦП є *мікропроцесор* (МП), за допомогою якого здійснюється виконання арифметичних і логічних операцій над числами, обробка текстової і графічної інформації, передача даних між регістрами й осередками ОЗП і т.д. Відповідно до розрядності внутрішніх регістрів МП і побудовані на їхній основі ПЕОМ бувають 8-, 16-, 32- чи 64-розрядними.

Якщо ЦП здійснює і координує процес обробки даних, то ОЗП виконує функції їхнього збереження. Довжина слова інформації, що міститься в ОЗП, звичайно 1 байт (8 біт). На сьогодні обсяг ОЗП досягає сотень Мбайтів.

Крім перерахованих головних компонентів, до складу системного блоку входять *адаптери (контролери)* периферійних пристроїв: дисплея, накопичувача на дисках, принтера. Вони виконують функції керування,

передачі і прийому необхідної інформації, наприклад забезпечують її запис і читання на гнучкому магнітному диску. Дані між різними компонентами передаються по системній шині, що забезпечує універсальний інтерфейс зв'язку між ними. *Інтерфейс* – це сукупність уніфікованих технічних і програмних засобів, необхідних для підключення пристроїв до тієї чи іншої системи.

Клавіатура служить для введення інформації в ПЕОМ. Вона має вигляд панелі з клавішами для набору числової, текстової і графічної інформації.

Клавіатура є головним засобом введення інформації в ПЕОМ. Знання її особливостей необхідно для роботи з машиною. У різних клавіатурах ПЕОМ клавіші поділяються на такі групи:

1) алфавітно-цифрові і знакові (із символами А,..., Z, а, ..., z, А,..., Я, а,..., я, 0,...,9, + ,-,*,/,&, j ,; , : ,\ ,?,!,., # і т.п.), а також спеціальні Esc (Ключ) , Tab (Табуляція), Enter (Уведення) , Back Space (Видалення) ;

2) функціональні - F1,...,F 12 (Ф1,...,Ф12) ;

3) службові для керування переміщенням курсору і редагуванням - Up (t), Down (I) , Left (<-), Right (->), Home (На початок), End (На кінець), Ins (Вставка), PageUp (fi), PageDn (9), Del (Видалення);

4) службові для зміни регістрів і модифікації кодів інших клавіш-Alt (Додаткові), Ctrl (Керувальні), Shift (Прописні);

• 5) службові для фіксації регістрів — Caps Lock(ФПБ), Scroll Lock (ФСД), Num Lock (Цифрові);

6) допоміжні— Print Screen (Друк) і ін.

Розрізняють три рівні подання й обробки сигналів, що надходять із клавіатури, – фізичний, логічний і функціональний.

На *фізичному рівні* аналізуються сигнали, що надходять у системний блок при натисканні і відпусканні клавіші. Наприклад, клавіатура ПЕОМ IBM/PC – це невеликий самостійний комп'ютер. У ній розташовано свій МП Intel 8048, що стежить за натисканнями клавіш і передає їхній стан. В основі кодування переданих у системний блок сигналів лежить звичайна нумерація клавіш. При натисканні кожної з них у системний блок посилається код, що відповідає її порядковому номеру – Скен-код. При відпусканні клавіші теж генерується її номер, збільшений на 128 (додатковий Скен-код). МП Intel 8048 дозволяє буферизувати до 20 натискань, якщо ЦП не може прийняти відповідні коди. При натисканні клавіші довше, ніж на 0,5с автоматично генерується послідовність її основних Скен-кодів з частотою 10 разів на секунду. Це імітує серію дуже швидких натискань.

Логічний рівень підтримується базовою системою введення-виведення (BIOS) через переривання з номером 9. Запит на таке переривання виробляє процесор клавіатури після натискання або відпускання клавіші. У відповідь на переривання службова процедура

BIOS зчитує код сканування клавіш з порту клавіатури (порт 96), а потім пересилає в порт клавіатури команду очищення буфера МП Intel 8048. Зчитаний код заноситься в буфер BIOS, що має об'єм 15 байт. Процедура BIOS перетворить отриманий із клавіатури Скен-код у спеціальний двійковий код. Молодший із двох байтів для клавіш першої групи містить ASCII-код, що відповідає зображеному на клавіші знаку. У старшому байті записаний вихідний Скен-код натиснутої клавіші. Якщо в першому байті записаний нуль, то відповідний код називають *розширеним ASCII-кодом.* У цьому випадку старший байт містить Скен-код натиснутої клавіші або він відповідає визначеній комбінації одночасно натиснутих клавіш. Це дозволяє перевірити той факт, що натиснута в даний момент клавіша не відноситься до групи 1. Інформація про стан клавіш Ins, Caps Lock, Num Lock, Scroll Lock, Alt, Ctrl, Shift (ліва і права клавіші окремо) зберігається в байтах ОЗП з адресами 1047 і 1048. Вона дозволяє визначити, які дії з клавіатурою зроблені (натиснута алфавітно-цифрова, спеціальна клавіша або використана комбінація).

На *функціональному рівні* окремим клавішам програмним шляхом передаються певні функції, що реалізуються при натисканні цих клавіш.

Залежно від того, до якої групи віднесена клавіша, розрізняються способи кодування, прийняті в ПЕОМ. Клавішам групи 1 відповідають прості ASCII-коди. Однак вони можуть модифікуватися залежно від того, чи натиснута в цей же момент клавіша з групи 4 або раніше – клавіша з групи 5. Клавішам груп 2 і 3 відповідають розширені ASCII-коди. Якщо разом з ними натиснути клавішу групи 4, то теж генеруються розширені ASCII-коди, але вони будуть іншими.

Клавішам груп 4 і 5 не відповідають ніякі коди. Вони лише впливають на модифікацію кодів попередніх груп. Додаткова інформація фіксується в байтах ОЗП з адресами 1047,1048.

Таким чином, набір кодів, які генеруються при різних комбінаціях клавіш, досить великий, що забезпечує користувачу великі можливості у керуванні ПЕОМ.

Дисплей є основним пристроєм для відображення, введеної і виведеної інформації. В процесі роботи користувач, натискаючи клавіші клавіатури, вводить у ПЕОМ цифри, букви, спеціальні знаки, що відразу ж з'являються на екрані дисплея. Розрізняють алфавітно-цифрові і графічні дисплеї з кольоровим чи монохромним зображенням.

Підтримка роботи дисплея здійснюється за допомогою відповідного адаптера (контролера). У ПЕОМ використовуються адаптери двох типів: *одноколірний (монохромний)* і *кольоровий графічний.* Зображення на екрані формується в двох основних режимах – текстовому і графічному. У першому з них на екран виводяться тільки символи, у другому можна будувати складні довільні зображення. Адаптер пов'язує ЦП із дисплеєм за допомогою пристрою, що має назву *контролер електронно-променевої*

трубки (екрана). Він має також програмувальні порти введення-виведення, знакогенератор і оперативну пам'ять, що зберігає виведену інформацію.

Образ екрана зберігається в оперативній пам'яті. Розмір екрана для текстового формату складає 25 рядків по 80 знаків або 25 рядків по 40 знаків. Для кодування кожного знака використовується матриця розміром 8×8 точок. Будь-якому відображуваному символу відповідають два байти екранної пам'яті, що встановлена безпосередньо в *дисплейному адаптері* і входить до загального адресного простору ПЕОМ. Байт із парною адресою специфікує символ (містить його ASCII-код). Дані з цього байта використовуються для відображення символу за допомогою знакогенератора. Програма знакогенератора зберігається в постійному запам'ятовувальному пристрої (ПЗП) дисплейного адаптера. Байт із непарною адресою специфікує режим відображення (колір, яскравість, мерехтіння і т.д.) – це так званий *байт атрибутів*.

При графічному режимі елементом зображення є точка чи піксель. Мінімальний об'єм пам'яті, використовуваний кольоровим графічним адаптером, звичайно складає до 32 байтів. Це дозволяє відображати 1280×1024 точок у кольоровому режимі. Зазвичай в САПР використовуються спеціальні дисплеї з великими розмірами і великим числом відображуваних точок.

Кожній точці екрана монохромного графічного дисплея відповідає біт інформації в екранній пам'яті. Якщо в ньому міститься одиниця, то точка висвічується, якщо нуль – ні. Для того, щоб структура збереженого відбитку екрана відповідала механізму сканування променями, пам'ять дисплея організована у вигляді двох блоків. Якщо об'єм пам'яті складає до 32 байтів, блоки зміщені один відносно іншого на 16 байтів. Перший блок містить інформацію про точки парних рядків, другий – непарних рядків. На початку першого блоку пам'яті розташовуються біти для нульового рядка на екрані. Їх 640, що дорівнює 80 байтам. Наступні 80 байт відповідають не першому, а другому рядку екрана, далі – четвертому, шостому і т.д. Першому рядку екрана відповідають 80 перших байт другого блоку. Адреса першого байта другого блоку відстає на 8192 байту від адреси першого байта першого блоку. В другому блоці зберігається інформація для непарних рядків екрана (першого, третього, п'ятого і т.п.). Зазначену особливість необхідно враховувати при роботі з екранною пам'яттю.

Розрізняють графічний і псевдографічний режими відображення. Як уже відзначалося, у *графічному режимі* елементарною одиницею інформації на екрані є точка. Зображення збирається з різних точок, при цьому можуть мінятися їхній колір і інтенсивність світіння. У *псевдографічному режимі* картинка на екрані формується зі спеціальних знаків, відображуваних у матриці розміром 8×8 точок. Ці знаки кодуються розширеними ASCII-кодами (діапазон від 128 до 255). Приблизно третя

частина цих кодів призначена для подання креслень і малюнків (інші – для символів національних алфавітів і т.п.). Псевдографічні елементи – це лінії, прямокутники, кружки і т.п. Зображення з них збирається так само, як з мозаїки. Один з векторів переривання (переривання з номером 31) використовується для указання на розширену таблицю ASCII. У результаті її можна змінювати і добудовувати за власними потребами. Псевдографічний режим має перевагу в тому, що витрачається менший (приблизно в 4 рази) об'єм екранної пам'яті. Для екрана 8×8 точок у графічному режимі потрібно 8 байтів, а в псевдографічному режимі — тільки 2 (байт символу і байт атрибутів).

Обслуговування відеомоніторів нижчого рівня здійснюється через переривання BIOS з номером 16. Користувачу пропонуються 19 різних функцій обслуговування, наприклад: установка режиму роботи дисплея (текстовий 40-позиційний, текстовий 80-позиційний, графічний 320×200 точок, графічний 640×200 точок, монохромний і т.п.), вибір розміру курсора, введення і висновок символів, установка атрибутів, переміщення курсора, визначення кольору, задання активної сторінки екрана і т.п.

Сторінковий режим використовується при роботі кольорових графічних дисплеїв у текстовому режимі для ефективного відтворення динамічних зображень. Сторінка є образом екрана в пам'яті комп'ютера. У 80-позиційному текстовому режимі кольоровий графічний адаптер вимагає пам'яті в чотири рази (а в 40-позиційному текстовому режимі в вісім разів) менше фактично наявної. Тому пам'ять такого адаптера може зберігати чотири чи вісім сторінок. Сторінка є повною копією екрана. Інформація, що зберігається в кожній з них, може бути в будь-який момент часу відображена. Одержавши команду, дисплейний адаптер переключається з однієї сторінки на іншу, миттєво оновлюючи вміст екрана.

Накопичувачі на гнучких і жорстких дисках служать для організації зовнішньої пам'яті, необхідної для тривалого збереження великих масивів інформації, що не руйнується при вимкненні напруги живлення ПЕОМ. Обсяг інформації на одному гнучкому магнітному диску складає від 1,44 до 50 Мбайтів, а на вінчестері – до сотень Гбайтів.

Гнучкий магнітний диск (ГМД) – це кругла пластина гнучкого пластику, покрита магнітним матеріалом. Пластина поміщена в квадратний запобіжний корпус, що має чотири отвори. Центральний отвір призначений для захоплення диска приводом дисковода. Подовгуватий отвір необхідний для доступу головок записування-зчитування. Отвір у корпусі дозволяє знаходити індексний отвір на диску, що використовується для ініціалізації початку і кінця доріжки. Квадратний проріз на краю корпуса призначений для захисту від записування. Якщо він відкритий, записування дозволено; якщо закритий – заборонено.

Дані на дискеті розміщуються на замкнутих концентричних доріжках, які розташовані на певній відстані від центрального отвору.

Доріжки поділяються на частини, що називаються *секторами.* Сектори є основною одиницею збереження інформації на дискеті. При звертанні до НМД завжди записується чи зчитується цілий сектор, незалежно від обсягу запитуваної інформації.

Дані можуть розташовуватися як на одній стороні дискети, так і на обох. Ці сторони нумеруються цифрами 0 і 1. Кожна з них має 40 чи 80 доріжок з номерами від 0 до 39 або від 0 до 79. Доріжка містить 8 чи 9 секторів з номерами від 1 до 8 чи від 1 до 9. Нульова доріжка розташовується ближче до зовнішнього краю дискети. У двосторонніх дискет за останнім сектором нульової сторони знаходиться перший сектор тієї ж доріжки, але першої сторони. Як при односторонньому, так і при двосторонньому форматі за останнім сектором однієї доріжки йде перший сектор наступної доріжки. Важливою характеристикою диска є *щільність запису.* При записі 40 доріжок даних на поверхні диска щільність вважається *подвійною,* а 80 доріжок — *почетвереною.* Подвійна щільність складає 48 доріжок на дюйм (дюйм дорівнює 25,4 мм), а почетверена - 96 доріжок на дюйм. Розмір сектора для диска діаметром 133,4 мм, підтримуваний BIOS, може складати 128, 256, 512 і 1024 байта (майже завжди використовується розмір 512 байтів).

Є два способи розмітки доріжок на сектори: *фіксований (апаратний)* і *програмний.* Для машин сімейства IBM/PC розташування доріжок на диску і число сторін визначаються характеристиками НМД і є незмінними. Однак кількість секторів на доріжці і їхній розмір можна задати програмно в процесі форматування.

Комп'ютери сімейства PS-2 використовують переважно 3,5-дюймові плоскі жорсткі диски, що дозволяють зберігати до 1,44 Мбайтів. Їхня товщина приблизно така сама, як у двох гнучких дисків, а діаметр у 1,5 рази менший.

Накопичувачі на жорстких магнітних дисках (вінчестери) характеризуються власними форматами даних. Будь-який такий диск має *фізичний* формат, визначений розмір сектора (у байтах), число секторів у циліндрі (у жорстких дисках замість доріжок розглядаються циліндри), число циліндрів і сторін. *Алогічний формат,* що характеризує спосіб організації інформації (наприклад, диск має об'єм 20 Мбайтів-4 сторони, 17 секторів, 615 циліндрів).

Жорсткі диски надходять до споживача, маючи певний фізичний формат, що встановлюється в процесі виготовлення диска. Логічний формат твердого диска до його використання операційною системою задається за два етапи. Спочатку диск поділяється на частини, кожну з яких дозволяється використовувати своєю операційною системою. Потім кожна частина форматується відповідно до вимог тієї системи, для якої вона призначена. Наприклад, фізичний диск об'ємом 20 Гбайтів можна розділити на 2 логічних диски кожний об'ємом 10 Гбайтів.

Принтери призначені для виведення інформації з ПЕОМ на папір. Звичайно використовуються принтери з друкувальними головками, у яких формоване зображення будується з дрібних точок, що утворюються в результаті ударів голок по паперу через барвну стрічку. У популярних принтерах RAVI 801 OM, EPSON FX-85, FX-800/1000 друкувальна головка містить 9 вертикально розташованих голок. Ці принтери мають наступні характеристики: швидкість друку – 40, 160,200,240 знаків у секунду (залежно від шрифту, якості друку); ширина паперу – від 254 до 406 мм; набори знаків і формат друку – стандартно встановлені і запрограмовані.

Для підключення інших типів периферійних пристроїв (графопобудовників, пристроїв знімання координат, маніпуляторів типу "миша" і т.п.) передбачено два види інтерфейсів: паралельний і послідовний. *Паралельний інтерфейс* використовується для паралельної передачі байтів (до відповідного каналу такого типу підключається принтер). *Послідовний інтерфейс* забезпечує передачу даних у послідовному коді зі швидкостями 110, 150, 300, 600, 1200, 2400, 4800 чи 9600 бод. Допустимі режими синхронного й асинхронного зв'язку.

Контрольні запитання

1. Технічне забезпечення (ТЗаб) САПР.
2. Структура комплексу ТЗаб.
3. Принципи дії пристроїв відображення графічної інформації.
4. Принципи квантування відеосигналу.
5. Принципи дискретизації графічної інформації.
6. Класифікація пристроїв вводу відеоінформації.
7. Використання лінійних та матричних фотоприймачів для вводу відеоінформації.
8. Структура автоматизованного робочого місця проектувальника.
9. Персональні ЕОМ в САПР.
10. Організація АРМ на базі персонального комп'ютера.

5. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ САПР

Програмне забезпечення (ПЗ) САПР – це сукупність всіх програм і експлуатаційної документації до них, необхідної для виконання автоматизованого проектування.

Програмне забезпечення займає особливе місце в САПР, тому що в програмах реалізуються методи автоматизованого проектування. Програмне забезпечення САПР відноситься до складних програмних систем; на розробку ПЗ затрачається до 80÷90% коштів, що виділяються на створення САПР. При проектуванні ПЗ САПР закладаються його характеристики, визначається зміст і характер робіт на всіх етапах створення ПЗ. Вимоги, які висуваються на етапі проектування ПЗ істотно впливають на основні експлуатаційні характеристики САПР в цілому.

5.1 Склад програмного забезпечення САПР

Програмне забезпечення САПР включає програми на машинних носіях, тексти програм, інструкції з їх використанням та іншу програмну документацію.

ПЗ САПР поділяється на загальне і спеціальне.

1. *Загальне* ПЗ САПР містить програми керування, контролю і планування обчислювального процесу, розподілу часу і пам'яті ЕОМ. Функції загального ПЗ виконує операційна система. Загальне ПЗ є інваріантним до об'єктів проектування. Основні функції загальносистемного ПЗ:

- управління процесом, введення, виведення і обробки інструкцій користувачів;
- діалоговий взаємозв'язок з користувачами в процесі проектування;
- зберігання, пошук, аналіз, модифікація даних;
- захист їх цілісності;
- розв'язання загальносистемних задач;
- контроль і діагностика в процесі розв'язання задач проектування.

До складу загального ПЗ входять: моніторна діалогова система, система управління базами даних; інформаційно-пошукові системи; геометричні і графічні процесори, засоби формування графічної і текстової інформації, засоби для виконання загальнотехнічних розрахунків.

2. Спеціальне ПЗ (СПЗ) функціонує в операційному середовищі, яке складається з загальносистемного і базового ПЗ. Основною функцією СПЗ САПР є отримання проектних рішень. СПЗ САПР створюється з врахуванням організації і можливостей загального ПЗ.

5.2 Основні компоненти ПЗ САПР

Варіанти організації ПЗ САПР різноманітні і залежать від багатьох факторів, головними з який є:

- предметна галузь, аспекти й рівні описів проектованих об'єктів, що створюються за допомогою ПЗ;
- ступінь автоматизації окремих проектних операцій і процедур;
- архітектура і склад технічних засобів, режим їх функціонування;
- ресурси, витрачені на розробку ПЗ.

Як основний розглянемо варіант організації ПЗ однорівневої САПР. Програмне забезпечення САПР поділяється на дві складові частини, що використовуються при проектуванні і обслуговуванні підсистем САПР. Надалі вони іменуються підсистемами ПЗ.

До обслуговувальних підсистем ПЗ відносяться:

- діалогова ДП;
- керування базами даних СУБД;
- інструментальна ІП;
- монітор, що забезпечує взаємодію всіх інших підсистем і керування їх виконанням.

Діалогова підсистема ПЗ організовує інтерактивну взаємодію користувача САПР із керувальною підсистемою ПЗ, підготовку і редагування вихідних даних, перегляд результатів роботи проектних підсистем, які функціонують у пакетному режимі.

Підсистема керування базами даних СУБД реалізує однаковий доступ до загальної бази даних САПР і до індивідуальних БД користувачів. Призначення БД таке:

- збереження даних нормативно-довідкового характеру (уніфіковані вироби, типові проектні рішення, раніше виконані розробки і т.д.);
- збереження результатів виконаних етапів поточного проекту;
- забезпечення інформаційної сумісності різних підсистем САПР.

Інструментальна підсистема програмування, основу якої складає генератор прикладних програм, що синтезує нові програми з уніфікованих модулів і підпрограм, розроблених користувачем, необхідна для досягнення відкритості ПЗ САПР. Генератор прикладних програм містить у собі також засоби автоматичної розробки трансляторів для вхідних мов підсистем, що проектують САПР.

Підсистеми, що проектують ПЗ, можуть бути об'єктно-залежними (проблемно-орієнтованими) чи об'єктно-незалежними (методо-орієнтованими, інваріантними). Об'єктно-незалежні підсистеми ПЗ орієнтовані на розв'язання задач проектування при наявності їх попередньо виконаної математичної постановки (наприклад, підсистеми параметричної оптимізації, розв'язання систем рівнянь у частинних

похідних, систем звичайних диференціальних рівнянь та ін.). Використання об'єктно-незалежних підсистем ПЗ менш ефективне, оскільки в них не враховується специфіка задач конкретної предметної галузі і потрібна досить висока математична підготовка користувача. Такі підсистеми призначені для розв'язання задач, для яких у складі САПР відсутні відповідні проблемно-орієнтовані підсистеми. Крім того, вони складають основу для генерації проблемно-орієнтованих підсистем ПЗ.

Підсистемами ПЗ можуть бути *прості програми*, орієнтовані на вузький клас об'єктів, і прості аналітичні моделі. Але частіше проектують підсистеми ПЗ, які є універсальними *пакетами прикладних програм* складної структури, що володіють своїми моніторами, локальними базами даних і засобами їхнього управління, тому нижче поряд з терміном «проектувальна підсистема ПЗ» будемо використовувати й інший термін – «проектувальний пакет ПЗ». Деякі з таких пакетів можуть реалізовувати не тільки окремі проектні операції і процедури, а й допускати множинний доступ (тобто роботу одночасно з декількома користувачами), в останньому випадку вони повинні мати свої локальні засоби підтримки діалогової взаємодії.

Підсистема інтерактивної машинної графіки (ПМГ) займає проміжне положення між підсистемами ПЗ. З одного боку, засоби машинної графіки обслуговують ряд проектувальних підсистем, що проектують (зазвичай, це пакети функціонального проектування), де вони використовуються, в основному, для наочного подання вхідної і вихідної інформації (у вигляді схем, часових діаграм, гістограм і т.д.). З іншого боку, у багатьох підсистем конструкторського проектування ПЗ інтерактивної машинної графіки входить як основна частина. Тому в САПР можлива наявність декількох пакетів машинної графіки (базового, як обслуговувального, й одного чи більше – у складі підсистем конструювання).

Характерна риса розглянутої архітектури ПЗ САПР – строга розмежованість проектувальних і обслуговувальних підсистем. Такий підхід до побудови ПЗ забезпечує, по-перше, його легке розширення і модифікацію, по-друге, перенесення на інші технічні засоби, оскільки всі машинно-залежні програмні компоненти локалізовані в обслуговувальних підсистемах. Недоліком зосередження обслуговувальних функцій в окремих підсистемах є складність в організації до них множинного доступу.

Програмне забезпечення САПР орієнтоване на роздільне редагування всіх його підсистем і їхнє динамічне завантаження в оперативну пам'ять (ОП) за потребою. Монітор і діалогова підсистема ПЗ резидентні, тобто постійно знаходяться в ОП. У суміжну з ними зону динамічно завантажуються обслуговувальні і проектувальні підсистеми, що проектують при цьому обслуговувальні підсистеми займають ділянки пам'яті з меншими адресами. Зона пам'яті, що залишилася не зайнятою,

може бути використана для розміщення даних. Динамічна структура ПЗ порівняно з оверлейною структурою, що вимагає спільного редагування всіх підсистем ПЗ, характеризується легкістю розширення і модифікації, а також значною економією ОП. Однак, для динамічної структури необхідні додаткові витрати на організацію взаємодії проектувальних і обслуговувальних підсистем.

Монітор САПР. Керування ходом обчислювального процесу і координація взаємодії підсистем САПР здійснюються монітором. Ті самі задачі, але в рамках окремих пакетів вирішуються моніторами цих пакетів. До функцій моніторів входять:

- прийом і інтерпретація направлених до них команд користувача;
- завантаження й активізація компонентів ПЗ, організація маршрутів їхнього виконання;
- встановлення взаємодії між підсистемами;
- динамічний розподіл пам'яті;
- обробка переривань від дисплея користувача;
- сервісні функції (реєстрація користувачів, збір статистики, ведення служби часу, обробка збоїв і т.д).

Мова керування монітором САПР досить проста, в її основі лежать команди виклику необхідних підсистем, що проектують ПЗ, і задання їм керувальних параметрів, а також команди, що описують спосіб інформаційного обміну між підсистемами – через оперативну чи зовнішню пам'ять, за допомогою підсистеми управління БД. Засоби цієї мови повинні дозволяти створювати макрокоманди, що визначають маршрути виконання підсистем, що проектують ПЗ. Мови керування проектувальних пакетів значно складніші, оскільки вони повинні відображати всі можливі постановки задач проектування в конкретних предметних галузях, розв'язання яких допускають пакети. Звичайно, ці мови мають процедурний характер.

У загальному випадку завантажені підсистеми, що проектують ПЗ, можуть функціонувати або як звичайні підпрограми, підлеглі керувальній підсистемі ПЗ, або як «паралельно» виконувані підзадачі, які здатні змагатися між собою і монітором за керування. Функціонування декількох пакетів одночасно як підзадачі є виправданим тільки у випадках, коли кожний з них окремо не здатний завантажити процесор ЕОМ і «розпаралелювання» не позначається на ефективності та зручності роботи кожного з користувачів. Очевидно, що при цьому кожна з підсистем, що проектують ПЗ, повинна мати свою локальну підсистему діалогової взаємодії. Створення підзадач – один зі способів забезпечення множинного доступу користувачів до САПР, однак його реалізація значно ускладнює керувальну підсистему: по-перше, виникає задача динамічного розподілу ресурсів ЕОМ; по-друге, з'являється потреба в механізмі, що дозволяє будь-яким чином усувати конфлікти в роботі підзадач. Такі конфлікти

можуть виникнути, наприклад, при одночасному звертанні декількох проектованих пакетів до підсистеми керування базою даних. Конфлікти можуть бути усунуті використанням черг запитів до СУБД, у яких запити на обслуговування підсистем ПЗ базою даних розташовуються в порядку надходження і пріоритетності.

Для забезпечення доступу до однієї проектувальної підсистеми декількох користувачів може бути використаний економічніший спосіб, заснований на реалізації основних програмних компонентів пакета.

Взаємодія підсистем. Взаємодія керувальної підсистеми ПЗ і моніторів пакетів проектування здійснюється через стандартний інтерфейс, що є формальними правилами передачі фактичних параметрів. У підсистеми, що проектують ПЗ, передаються:

- параметри, що задають режим функціонування;
- адреси точок входу в обслуговувальні підсистеми ПЗ;
- адреси динамічно розподілених областей пам'яті, призначених для інформаційного обміну між різними підсистемами ПЗ.

Кожен пакет, що входить до складу САПР, має паспорт, що зберігається в базі даних САПР. Паспорт містить такі дані про пакет:

- розмір займаної області ОП;
- імена необхідних обслуговувальних підсистем ПЗ;
- імена режимних параметрів і їхнього значення за замовчуванням;
- ім'я мови програмування, у стандарті якої пакет використовує подання структур даних;
- місцезнаходження в пакеті оброблювача переривань від дисплея користувача (якщо він передбачений);
- покажчики на можливі способи обміну інформацією з іншими проектувальними підсистемами ПЗ (через ОП, базу даних чи файлову систему ЕОМ) і т.д.

Монітор САПР, одержавши команду про активізацію будь-якої проектувальної підсистеми ПЗ, зчитує з бази даних її паспорт, перевіряє коректність команди і можливість завантаження підсистеми. Потім він поміщає в ОП необхідні обслуговувальні підсистеми ПЗ (якщо їх там ще немає), слідом за ними – необхідну проектувальну підсистему, а потім у строгій відповідності з даними з паспорта будується звертання до цієї підсистеми. Після закінчення роботи підсистеми вона вивантажується з ОП.

Динамічне завантаження проектувальних підсистем не дозволяє прямо звертатися з них в обслуговувальні підсистеми. Тому для організації зв'язку, в проектувальних підсистемах, передаються адреси точок входу всіх необхідних обслуговувальних підсистем.

Деякі проектувальні підсистеми ПЗ для розв'язання задач високої розмірності вимагають великих витрат машинного часу й ОП, наприклад:

задачі аналізу складних динамічних об'єктів, їх параметрична оптимізація; синтез тестів для цифрових пристроїв трасування друкованих плат і т.д. Використання інтерактивного режиму на етапі розв'язання таких задач недоцільно, але він необхідний на підготовчих стадіях і при інтерпретації результатів. Для таких випадків у складі ПЗ САПР необхідно мати обслуговувальну підсистему утворення фонових завдань. Якщо САПР функціонує на обчислювальній установці, що має зв'язок з іншими ЕОМ, то така підсистема повинна забезпечувати можливість передачі фонових завдань на одну з цих ЕОМ. Після завершення фонового завдання, його результати можуть бути переглянуті й оброблені користувачем за допомогою засобів підсистеми ПЗ, що створила це завдання.

Важливою функцією керувальної підсистеми САПР і моніторів пакетів є те, що динамічний розподіл ОП потрібен завжди, коли пакет призначений для роботи з даними різного об'єму. Монітор САПР виділяє ОП для забезпечення інформаційного обміну між підсистемами; монітори пакетів роблять це за запитами модулів пакета, якщо засоби мови, що використовувалися для їхнього програмування, недостатні для ефективного використання ОП.

Засоби динамічного розподілу пам'яті – обов'язкові компоненти всіх сучасних операційних систем і мають місце в багатьох мовах програмування (за винятком мов ФОРТРАН і КОБОЛ). *Для ефективної роботи колективу користувачів необхідний множинний доступ до САПР.* Вище були розглянуті два способи організації одночасної роботи декількох користувачів, однак для обох способів характерним є те, що активізація завдань користувачів відбувається в хаотичному порядку і на невизначений час. Цю проблему вирішує режим розподілу часу, але його реалізація засобами прикладного ПЗ САПР є складною задачею. Більш доцільна реалізація режиму розподілу часу за допомогою загального програмного забезпечення – відповідних ОС ЕОМ.

Розглянутий варіант архітектури ПЗ САПР порівняно простий, він придатний для створення САПР середніх розмірів. Великі промислові САПР, що функціонують на мережах ЕОМ, мають складні, розподілені по ЕОМ монітори, спеціальні обслуговувальні підсистеми інформаційного обміну, керування технологічним обладнанням, планування і керування ходом проекту. Такі САПР інтегровані з автоматизованими системами наукових досліджень, технологічної підготовки виробництва, з гнучкими автоматизованими виробництвами. Їх ПЗ відбиває специфіку конкретних предметних галузей, прийняті в них маршрути проектування і структуру наявних на підприємстві технічних засобів.

ПЗ САПР відноситься до самих складних програмних комплексів. Воно є закінченим програмним продуктом, що поставляється як промисловий виріб.

5.3 Спеціальне програмне забезпечення САПР

Спеціальне програмне забезпечення (СПЗ) САПР – це сукупність цільових програмних засобів, за допомогою яких реалізуються різні етапи процесу автоматизованого проектування.

Склад і функції цього забезпечення залежать від об'єкта проектування, виду проектно-конструкторської та технологічної документації, архітектури технічних засобів.

Виходячи із загального визначення поняття інформації, проектування розглядають як процес перетворення початкового опису об'єкта в кінцевий. При автоматизованому проектуванні такий опис виконується в ЕОМ у вигляді цифрової моделі об'єкта проектування (ЦМО), що містить дані про структуру об'єкта проектування, його елементи та характеристики. На початку ЦМО формується на основі опису об'єкта проектування вихідною мовою. В процесі автоматизованого проектування ЦМО змінюється як структурно, так і за змістом інформації. Ці зміни виконує проектувальник або відповідні спеціалізовані програми автоматизованого проектування.

В машинобудівному проектуванні спеціальні програмні засоби забезпечують побудову математичної моделі технічних об'єктів і їх частин, проведення оптимізаційних і конструкторських розрахунків, введення і відображення описів в алфавітно-цифровій і графічній формах, формування і випуск конструкторської і технологічної документації, підготовку носіїв для управління автоматизованими технологічними процесами і інше.

В галузі радіоелектроніки спеціалізовані програми забезпечують логічне і функціональне моделювання схем, проектування фотошаблонів інтегральних схем, проектування друкованих плат, випуск конструкторської документації і підготовку носіїв для виготовлення друкованих плат.

До складу спеціального ПЗ входять тисячі програм, об'єм яких складає мільйони машинних команд. Створення таких великих програмних систем пов'язане з вирішенням проблем декомпозиції СПЗ на підсистеми, розробки інтерфейсів між підсистемами, вибором архітектури операційної системи залежно від конфігурації технічних засобів, можливостей операційної системи і базового програмного забезпечення. На рисунку 5.1 показано варіант загальної архітектури СПЗ однорівневої САПР.

До обслуговувальних підсистем програмного забезпечення відносяться: діалогова (ДП), управління базами даних, інструментальні (ІП), а також монітор, що забезпечує взаємодію всіх інших підсистем і керування їх виконанням. Проектувальні підсистеми (ПП) можуть бути виконані у вигляді прикладних програм або їх сукупності зі своїми моніторами.

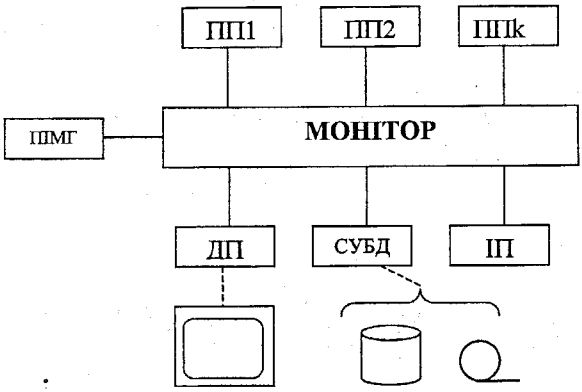


Рисунок 5.1 – Варіант архітектури СПЗ

$ПП_i$ – *i*-а проектувальна підсистема; ДП – діалогова підсистема; СУБД – система управління базами даних; ППГМ – система інтерактивної машинної графіки.

Незмінною частиною сучасних САПР є підсистема інтерактивної машинної графіки, яка забезпечує подання результатів проектування у вигляді креслень, схем, графіків, діаграм і діалоговий графічний режим взаємодії проектувальника з системою на етапах введення і коректування ЦМО, аналізу проміжкових і кінцевих результатів проектування, редагування креслень, що підготовлені системою з метою доведення їх до стопроцентної готовності і т.п.

Використання СУБД дозволяє:

- зменшити затрати на формування і підтримку нормативної довідникової бази про норми типових проектних рішень і т. д.;
- забезпечити інформаційну єдність роботи різних підсистем;
- контролювати доступ до даних із забезпечення захисту від несанкціонованого доступу, що підвищує ефективність, надійність, якість і гнучкість функціонування САПР.

СУБД забезпечує також реалізацію принципу логічної і фізичної незалежності даних. Включення нових структур даних виконується оперативно, без додаткових змін в раніше створених програмних засобах, що відповідає принципу розвитку системи.

Спеціальне ПЗ САПР поділяється на *прикладне* і *базове*.

5.3.1 Прикладне СПЗ САПР

Прикладне СПЗ реалізує алгоритми розв'язання проектних задач. При його розробці враховують склад математичного забезпечення САПР,

мови програмування, організацію інформаційної взаємодії між програмними модулями, структури даних, конфігурацію технічних засобів і обрану ОС. Найчастіше прикладне ПЗ реалізують *пакетами прикладних програм*.

Пакет прикладних програм – сукупність програм які сумісні між собою і забезпечують розв'язання задач з деякої галузі знань, яка має назву – *предметна*.

У ППП виділяють *тіло* – частина, що залежить від предметної галузі і розв'язуваних задач, та *організувальну програму*, що забезпечує функціонування компонентів ППП. У ППП простої структури функції керування компонентами виконуються засобами ОС. Такі пакети простіше розробляти, але до складу ПЗ САПР вони включаються рідко. Звичайно проектуючи підсистеми САПР, реалізують ППП складної структури. Тут використовується монітор, що виконує функції організувальної програми. Тіло пакета складають програми, що здійснюють проектні операції і процедури.

Серед алгоритмів розв'язання задач проектування виділяють *евристичні* і *систематичні*.

Систематичні алгоритми базуються на строго обґрунтованих положеннях. Прикладом може служити метод логічного виведення формул математичної логіки з аксіом за фіксованими правилами виведення.

Евристичні алгоритми спираються на ідеї, підказані інтуїцією розроблювача. У таких алгоритмах немає строгого обґрунтування опорних положень, які називаються *евристиками*. В евристичних програмах не завжди ясні границі ефективного застосування, немає гарантій якості одержуваних рішень. При автоматизованому проектуванні, в основному, застосовуються евристичні алгоритми.

Для підвищення ефективності при розв'язанні проектних задач евристичні програми організовують в систему конкурентноздатних програм, що входять у тіло ППП. Вхідною мовою ППП користувач може викликати зміст, потрібну програму, змінити режим роботи і т.д.

ППП бувають *відкритими* і *закритими*.

У *закритих* пакетах є засоби для розв'язання будь-якої задачі з зазначеної предметної галузі. Наявність такого ППП рятує від необхідності писати власні програми для розв'язання проектних задач. Для цього досить знати методи проектування і вхідну мову ППП.

У *відкритих* ППП є можливість розвивати засоби розв'язання задач проектування в результаті конкретизації предметної галузі. До складу прикладного СПЗ САПР звичайно включають відкриті ППП, що забезпечує можливість модифікації системи.

5.3.2 Базове СПЗ САПР

Базове СПЗ САПР призначене для підтримки роботоздатності проектувальних підсистем. У його складі особливо слід виділити обслуговувальні підсистеми.

5.4 Основи розробки СПЗ САПР

Створення ПЗ САПР – складна науково-технічна задача, для розв'язання якої необхідні великі матеріальні витрати. Витрати на розробку складають велику долю (0,8÷0,90) всіх витрат на створення і експлуатацію САПР. Наприклад, розробка ПЗ САПР об'ємом до 500 тис. операторів мови програмування може вимагати сотні людино-років, тому що середня продуктивність праці програмістів в організаціях, що займаються розробкою ПЗ САПР, складає всього лише 1000÷2000 операторів/рік.

Вартість розробки ПЗ залежить від його об'єму, від використаних мов програмування.

> Під ***технологією програмування*** розуміється сукупність загальних і спеціалізованих знань про оптимальні засоби проведення процесу програмування, що забезпечує в заданих умовах отримання програмної продукції з заданими властивостями.

Програмне забезпечення САПР розробляється відповідно до основних принципів блочно-ієрархічного проектування складних систем – *модульності* (блочності) та *ієрархічності*. Принципи модульності, ієрархічності дозволяють організовувати колективну паралельну розробку різних частин ПЗ, створювати відкриті програмні системи, полегшувати їх комплексне налагодження і інформаційну сумісність.

5.4.1 Принцип модульності

> ***Модуль*** – структурна складова ПЗ, що розглядається як єдине ціле на визначених стадіях розробки або в процесі експлуатації, програмна одиниця, для створення якої потрібен мінімум знань про інші програмні одиниці, що складають ПЗ.

Перекомпонування і заміна модулів не повинні викликати перекомпонування ПЗ в цілому [8]. Таким чином, центральна концепція модульності – *незалежність.*

Модуль:

- повинен реалізовувати єдину функцію, мати один вхід і один вихід;
- повинен повертати керування тому, хто його викликав, мати можливість звертатися до інших модулів;
- не повинен зберігати історію своїх викликів;
- повинен бути порівняно невеликим (довжина вихідного тексту модуля, як рекомендується фахівцями - одна сторінка роздруківки) [9].

Для досягнення незалежності модулів часто при проектуванні ПЗ використовується принцип інформаційної локалізованості, який полягає в тому, що вся інформація про структуру даних багатьох компонентів ПЗ зосереджується («ховається») в одному модулі. Доступ до даних здійснюється тільки через цей модуль. Таким чином, конкретне подання структур даних "ховається" від усіх модулів-користувачів. При необхідності зміни структури даних усі пов'язані з цим модифікації будуть зосереджені тільки в одному модулі.

Подібно проектуванню, кодування (програмування) модулів ПЗ може здійснюватися в двох напрямках: зростаючому і спадному.

При *зростаючому* кодуванні після завершення проектування усього ПЗ приступають до кодування модулів самого нижнього рівня ієрархічної структури ПЗ, а потім, після їхньої перевірки, переходять до модулів вищого рівня і т.д., доки не буде виготовлена вся система. Недолік цього підходу в тому, що переконатися в правильності функціонування ПЗ, з погляду користувача, можна тільки після завершення кодування самого верхнього модуля.

Спадне кодування не має цього недоліку. У його найпростішій формі мається на увазі, що кодування починається після завершення проектування усього ПЗ. Але частіше під спадаючим кодуванням розуміється процес, що йде паралельно з проектуванням: після завершення проектування модулів якого-небудь рівня випливає їхнє кодування і тестування разом з модулями верхніх рівнів і тільки потім переходять на наступний нижчий рівень і т.д. Кодування зверху донизу має на увазі використання спадного тестування знову закодованих модулів із вже налагодженими модулями верхніх рівнів, при цьому ще не розроблені модулі нижчих рівнів імітуються спеціальними підпрограмами – *заглушками.* Тестуванням «останнього» модуля завершується комплексне тестування усього ПЗ. До основних переваг спадного тестування відносять:

- виключення тестування на рівні системи;
- тестування в першу чергу найважливіших інтерфейсів між модулями;
- ознайомлення з «макетом» майбутнього ПЗ користувача на відносно ранніх стадіях розробки [10].

Використання принципу покрокової деталізації при програмуванні окремих модулів ПЗ називають *структурним програмуванням.*

Мета структурного програмування – змусити програміста мислити ясно, писати програми мінімальної складності, полегшувати сприйняття програм [5]. Ця мета може бути досягнута, в першу чергу, за рахунок використання для вираження логіки програм невеликого набору простих структур керування: проходження, розгалуження і цикл. В результаті

иходить програма, що характеризується простотою, ясністю, легкою модифікованістю. Для програм, отриманих структурним методом, можна здійснити строгий і формальний доказ їхньої правильності з погляду виконання специфікацій.

5.4.2 Принцип ієрархічності

Виділяють такі ієрархічні рівні подання і, відповідно, проектування СПЗ САПР:

- системний;
- прикладних програм;
- підпрограм.

Системний рівень – конкретизуються функції програмно-методичного комплексу (ПМК), планується його структура і зміст, вибираються або розробляються мови проектування, встановлюється ступінь використання доступного для придбання готового загальносистемного і базового ПЗ, розробляються специфікації на окремі програми пакету.

Рівень прикладних програм – вибирається математичне забезпечення, розробляються специфічні алгоритми, встановлюється модульна структура програм, обираються структури даних, способи інформаційного інтерфейсу і мови програмування, розробляються специфікації на окремі програмні модулі.

Рівень підпрограм – виконується конкретизація типів і структур даних, здійснюється кодування алгоритмів, тобто їх запис обраною мовою програмування.

При розробці великих САПР можливе виділення додаткових проміжних рівнів.

5.4.3 Етапи проектування СПЗ

Процес проектування СПЗ складається із декількох етапів (рисунок 5.2). Етапи 1...4 відносяться до синтезу ПЗ і виконуються в спадній послідовності, а етапи 5...7 відносяться до налагодження і виконуються у зростаючій послідовності.

Етап 1. Аналіз вимог, запропонованих до ПЗ САПР. Основу цих вимог складають вимоги, перераховані вище. Вони доповнюються вимогами замовника, що включають просторово-часові обмеження, необхідні функції і можливості, режими функціонування, вимоги точності, надійності і т.д.

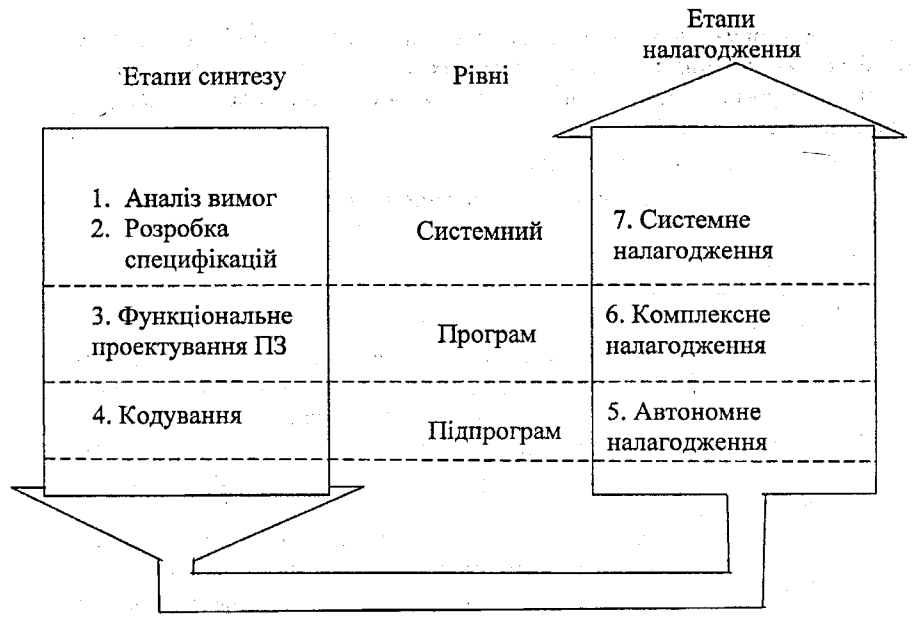


Рисунок 5.2 – Етапи проектування програмного забезпечення

Ця робота виконується групою системних аналітиків, кінцевим результатом діяльності яких повинен стати повний і несуперечливий опис вимог до проектовної системи мовою, зрозумілою розробникам ПЗ. На даному етапі:

- виявляються проектні процедури й операції, автоматизація яких можлива і доцільна;
- вивчаються особливості математичних моделей проектовних об'єктів;
- вибирається та розробляється математичне забезпечення;
- приймається рішення про типи ЕОМ і операційних систем, що будуть використовуватися;
- розглядається можливість використання готових компонентів ПЗ;
- розглядаються питання планування робіт, встановлюється їх черговість, етапність здачі підсистем САПР в експлуатацію.

Особлива увага приділяється дослідженню шляхів створення відкритого ПЗ.

Етап 2. Визначення специфікацій на СПЗ САПР. На цьому етапі:

- формулюється точний опис функцій САПР;
- розробляються і затверджуються вхідні і проміжні мови;
- форма вихідної інформації для кожної з підсистем;

- описується можлива взаємодія з іншими програмними комплексами;
- специфікуються засоби розширення і модифікації ПЗ;
- розробляються обслуговувальні і проектувальні інтерфейси підсистем;
- зважуються питання організації бази даних;
- затверджуються основні алгоритми, реалізовані в підсистемах.

Підсумком виконання даного етапу є експлуатаційні і функціональні специфікації, що містять конкретний опис СПЗ. Розробка специфікацій вимагає ретельної роботи системних аналітиків, що постійно контактують з користувачами, у більшості випадків нездатними самостійно сформулювати чіткі вимоги до системи.

Експлуатаційні специфікації містять інформацію про швидкодію СПЗ, витрати пам'яті, необхідні технічні засоби, надійність і т.д.

Функціональні специфікації визначають функції, які повинні виконувати САПР, тобто в них визначається що треба робити, але без вказівки, як це робити.

Специфікації повинні бути повними, точними і зрозумілими [6]. Повна специфікація виключає необхідність одержання розробниками СПЗ в процесі їхньої роботи від користувачів інших даних, крім тих, що є в специфікаціях. Точна специфікація не дозволяє тлумачити її в різних змістах. Ясна специфікація повинна бути легко написана і прочитана як користувачем СПЗ (можливо, через системного аналітика), так і його розробником, причому обоє вони повинні розуміти її абсолютно однозначно. Нижче будуть розглянуті деякі формальні засоби розробки специфікацій. Значення специфікацій при створенні СПЗ визначається трьома факторами:

- специфікації є завданням на розробку СПЗ, і строге їхнє виконання – закон для розробника;
- специфікації використовуються для перевірки готового СПЗ;
- специфікації, будучи невід'ємною частиною програмної документації, полегшують супровід і модифікацію СПЗ.

Завершується другий етап циклу життя підготовкою тестів, за якими будуть проводитися випробування при прийманні САПР. На підготовлені тестові дані надалі не буде впливати конкретна реалізація системи.

Етап 3. На цьому етапі формується структура СПЗ і розробляються алгоритми, що задаються специфікаціями. Встановлюється склад модулів з поділом їх на ієрархічні рівні на основі вивчення схем алгоритмів для типових задач проектування [7]. Обирається структура інформаційних масивів, що складають базу даних. Фіксуються міжмодульні інтерфейси.

Для ефективної реалізації даного етапу розробки СПЗ запропоновано методи, найвідоміші з яких будуть розглянуті нижче. Ці методи мають

одну загальну мету, реалізовану різними способами – ієрархічна розбивка складних задач створення СПЗ на підзадачі меншої складності. Результатом робіт на цьому етапі є специфікації на окремі модулі, подальша декомпозиція яких на підмодулі недоцільна.

Етап 4. На даному етапі відбувається програмування модулів будь-якою алгоритмічною мовою, тобто переклад розроблених алгоритмів на мову програмування. Цей етап менш складний, порівняно зі всіма іншими етапами циклу життя ПЗ, для його реалізації широко використовується метод структурного програмування. Одна з задач, яку необхідно розв'язати на даному етапі – обґрунтований вибір мов програмування.

Етапи 5, 6, 7. Налагодження виконується за допомогою тестів.

Тестуванням називається процес перевірки СПЗ (чи його компонентів), призначений для виявлення помилок у ПЗ. При цьому використовуються тестові набори, що включають у себе спеціально підібрані вихідні дані й еталонні результати.

Тестування містить у собі три кроки: автономне, комплексне і системне. При *автономному* тестуванні контролюється кожен компонент СПЗ окремо від інших.

В процесі *комплексного* тестування всього СПЗ виявляються помилки, пов'язані з порушенням специфікацій у всьому ПЗ САПР, при цьому використовуються тести, підготовлені на етапі 2.

Системне тестування – випробування ПЗ САПР на технічних засобах і даних користувача у виробничих умовах. Тестування є складовою частиною налагодження – процесу виявлення помилок, їх локалізації й усунення.

Необхідно відзначити, що надійність СПЗ закладається на ранніх етапах його циклу життя, правильне тестування дозволяє лише виявити більшість, з відносно невеликої кількості, помилок. Підвищити надійність погано спроектованих програм ніяке тестування не здатне.

Тестування – один з найменш формалізованих етапів циклу життя ПЗ. Можна автоматизувати сам процес тестування, але вибір тестових даних є творчою функцією розроблювачів СПЗ. Вартість і успішність тестування, а в кінцевому рахунку, і життєздатність усього СПЗ залежать від того, як їм вдається справитися з цією задачею. Тут подаються деякі аксіоми тестування:

Аксіома 1. Тестування повинно починатися з постановки мети.

Аксіома 2. Вважайте тестування ключовою задачею розробки.

Аксіома 3. Неможливо тестувати свою власну програму.

Аксіома 4. Гарним вважається той тест, для якого висока ймовірність знайти помилку, а не той, який демонструє правильну роботу програми

Аксіома 5. Готуйте тести як для правильних, так і для неправильних вхідних даних.

Етап супроводу. Найбільші витрати в циклі життя ПЗ доводяться саме на етап супроводу. Ці витрати для СПЗ САПР складаються, в основному, з витрат:

- на усунення помилок, не виявлених на етапі тестування;
- на внесення змін у первісну версію СПЗ, викликаних взаємним непорозумінням замовника і розробника СПЗ (в особі системних аналітиків) на перших двох етапах створення СПЗ, що призвело до розробки «не тих» специфікацій;
- на адаптацію вимог до СПЗ САПР, які швидко змінюються.

Усунення помилки під час експлуатації СПЗ обходиться, принаймні, в два рази дорожче, ніж на етапі тестування. Одна з проблем супроводу полягає в тому, що навіть для «правильно» спроектованого ПЗ виправлення однієї помилки спричиняє внесення нової помилки з імовірністю 0,2÷0,5. Третя складова витрат на супровід для СПЗ САПР найбільш істотна і пов'язана з задачею продовження терміну ефективної експлуатації САПР, оскільки зміна технологій промислового виробництва і розвиток математичного забезпечення АП відбуваються, зазвичай, більш швидкими темпами, ніж може створюватися СПЗ для нових САПР. Для уповільнення морального старіння СПЗ САПР пропонують три можливих способи:

- розробка САПР разом з розробкою нових промислових технологій;
- розробка САПР з урахуванням прогнозів на майбутнє;
- створення САПР, відкритих як до нових елементів математичного забезпечення, так і відносно нових технологій і предметних галузей.

Найперспективніший третій спосіб, хоча він і найскладніший у реалізації. Особливо гостро задача супроводу СПЗ стоїть для САПР, які експлуатуються у декількох організаціях, оскільки процес виправлення виявлених помилок, включення нових програмних компонентів різними користувачами дуже швидко призводить до появи декількох версій СПЗ, їх облік і контроль над ними стає неможливим. Щоб цього не відбувалося, усі корективи і доповнення необхідно вносити тільки у версію розроблювача СПЗ і передавати її всім користувачам одночасно через досить великі проміжки часу, які мають назву *періодом відновлення.*

Розглянута модель циклу життя ПЗ дуже умовна. Реальний процес розробки носить ітераційний характер, багато етапів перекривають один одного, виконуються паралельно.

5.5 Конкретні методи розробки ПЗ САПР

5.5.1 Методи формалізації розробки ПЗ

Останнім часом розробниками великих програмних систем усе більша увага надається створенню методів і засобів формалізації перших двох етапів циклу розробки ПЗ, оскільки помилки, внесені в проект на цих етапах, «найдорожчі». Дані методи і засоби дозволяють вручну чи автоматично виявляти й усувати такі недоліки функціональних специфікацій, як неповнота, суперечливість і неоднозначність. Приклад технології ручної розробки специфікацій на ПЗ даний К. Хенінджером у [6]. Суть цього підходу виражається в таких трьох принципах:

- Ставити питання раніше, ніж намагатися на них відповісти.
- Розділяти аспекти. Цей принцип забезпечує можливість кожному учаснику проекту зосередитися на добре визначеній сукупності питань і дозволяє легко модифікувати документ.
- Якнайбільше формалізації. З автоматизованих засобів аналізу вимог і документування специфікацій найвідоміші системи PSL/ PSA і SREM [6].

Система PSL/PSA дозволяє:

- описувати проектовані інформаційні системи у формі, придатній для обробки на ЕОМ, для чого використовується мова PSL;
- записувати і зберігати такий опис у базі даних;
- корегувати опис у базі даних;
- виконувати аналіз опису і видавати звіти.

Три останні функції реалізуються за допомогою пакета програм, що називається аналізатором формулювань задач PSA. Крім опису мовою PSL і коментарів у базі даних можуть зберігатися таблиці, масиви, матриці, а також графічні діаграми і схеми. У мову PSL навмисно не включені засоби процедурного типу, щоб не провокувати передчасний початок проектування ПЗ. Аналізатор формулювань задач PSA надає системним аналітикам звіти, які містять результати різного роду аналізів зведень, що є в базі даних.

Методологія розробки вимог до ПЗ, реалізована в системі SREM, схожа з PSL/PSA і включає в себе мову опису вимог RSL і підсистему REVS обробки і перевірки вимог. Мова RSL орієнтована на специфікацію необхідних кроків обробки даних у вигляді потокових графів. Підсистема REVS обробки і перевірки вимог складається з:

- транслятора з мови описів вимог RSL;
- центральної бази даних, що містить модель проектовної

програмної системи;

- автоматизованих засобів обробки інформації в базі даних.

Підсистема REVS має засоби машинної графіки, що дозволяють працювати з зображеннями потокових графів, а також забезпечує динамічне моделювання розроблюваного ПЗ, використовуючи для цього імітатори окремих компонентів ПЗ. Такі імітатори можуть застосовуватися для перевірки системних вимог і визначення набору алгоритмів, які найкраще відповідають цим вимогам.

Процес проектування ПЗ, як і будь-яких інших складних об'єктів – блочно-ієрархічний. Систематичне застосування декомпозиції дозволяє звести задачу великої розмірності до сукупності простих підзадач. Принципово можливі два підходи до проектування ПЗ: спадний і зростаючий.

5.5.2 Спадне проектування

Спадне проектування (покрокова деталізація) є послідовністю кроків, що уточнюють проект.

Перший крок – визначення способу розв'язання задачі загалом. За першим кроком випливають дрібні кроки в напрямку деталізації алгоритмів і структур даних. У ході цього процесу виділяються окремі модулі рішення і даних, і подальша конкретизація кожного модуля може розроблятися незалежно. Спеціально для реалізації стратегії спадного проектування розроблена мова проектування програм PDL [4]. Вона складається з двох частин:

- заданого набору операторів, побудованих за зразком тієї мови програмування, на якій планується вести кодування компонентів ПЗ;
- пропозицій природної мови.

Для опису логіки проектовної програми використовуються керувальні структури мови програмування (цикли, розгалуження, виклик підпрограм), а для опису даних і процедур їхньої обробки – природна мова.

Приклад спадного проектування з використанням мови PDL. Програма, структура якої подана на рисунку 5.3, може бути описана засобами мови PDL на першому кроці спадного проектування в такий спосіб:

монітор: ПРОЦЕДУРА;

виконати дію А; ЯКЩО (умова 1 виконується) ТО; виконати дію Б;

КІНЕЦЬ – ЯКЩО;

ЦИКЛ ПОКИ (умова 2 виконується); виконати дію В;

КІНЕЦЬ – ЦИКЛ;

КІНЕЦЬ – ПРОЦЕДУРА.

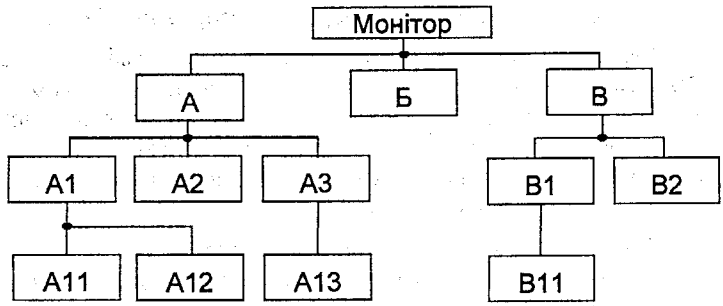


Рисунок 5.3 – Варіант структури програми

Наступним кроком деталізації може бути ухвалення рішення про виділення окремих підпрограм *А*, *Б* і *В*, що реалізують дії *А*, *Б* і *В*, і визначення даних, переданих у ці підпрограми. Крім того, з метою можливої економії ОП приймається, що підпрограма *Б* завантажується в неї динамічно. У результаті проект програми приймає такий вигляд:

монітор: ПРОЦЕДУРА;

визначити дані а, (3, у,…) підготувати а, р, у CALL *А* (а, р, у) ЯКЩО (умова 1 виконується) ТО;

завантажити в ОП підпрограму *Б*; CALL Б (у); КІНЕЦЬ – ЯКЩО; ЦИКЛ ПОКИ (умова 2 виконується);

CALL *У* (...) КІНЕЦЬ – ЦИКЛ;

КІНЕЦЬ – ПРОЦЕДУРА.

Після того, як прийнято рішення про те, що будь-яка частина поставленої задачі буде виконуватися окремим модулем, необхідно вичерпно і строго визначити функції цього модуля і його інтерфейс, не розглядаючи деталей його реалізації. Процедура покрокової деталізації монітора продовжується доти, доки на черговому кроці не стане очевидною його реалізацію операторами обраної мови програмування. Далі аналогічним чином проектуються модулі *А*, *Б* і *В* (можливо, паралельно різними програмістами). Наприкінці етапу проектування маємо логічну структуру ПЗ у вигляді дерева, як це показано на рисунку 5.3, детальні специфікації і проекти всіх його модулів.

Перевага спадного проектування полягає в тому, що воно дозволяє розробникам ПЗ зосередитися на основних, для даного моменту, проблемах і відкласти прийняття всіх тих рішень, що не повинні прийматися на даному етапі проектування. Для реалізації спадного проектування розроблено технології, найвідомішою з яких є методологія HIPO на основі графічних діаграм (фірма «IBM»). У цій методології

передбачено два види діаграм. За допомогою *ієрархічних діаграм* подається повна структура ПЗ. Кожен модуль цієї діаграми, у свою чергу, розкривається діаграмою «вхід – обробка – вихід» (ІРО – input – process – output).

Використання спадного проектування ставить перед розробниками ПЗ дві серйозні проблеми. Перша проблема пов'язана з початковим вибором алгоритмів і структур даних, реалізованих у ПЗ. Якщо, наприклад, вибір структури даних зроблений передчасно, то на нижчих рівнях проектування може виявитися її непристосованість для ефективної реалізації ряду алгоритмів – виникає необхідність перепроектування усього ПЗ. Тому спадне проектування вимагає із самого початку ставити і вирішувати найфундаментальніші задачі, відкладаючи частину питань для наступного розгляду. Друга проблема пов'язана з тим, що спадне проектування призводить до деревоподібної структури ПЗ, і цілком ймовірно, що в різних гілках цієї структури виявляться модулі зі схожими, але не однаковими специфікаціями. Звичайно ж, доцільно об'єднати специфікації і розробити один універсальний модуль, однак це призведе до перепроектування зухвалих модулів.

5.5.3 Зростаюче проектування

> ***Зростаюче проектування*** альтернативне методу покрокової деталізації. У той час як при спадному проектуванні первісна задача розбивається на підзадачі, які потім намагаються реалізувати, при зростаючому проектуванні, перш ніж приступити до вирішення задачі, створюються засоби, що дозволяють її вирішити.

Наприклад, якщо необхідно спроектувати зростаючим методом програму, аналогічну програмі, наведеній на рисунку 5.3, спочатку проектують модулі *A11* і *A12*, потім додають модуль *A1*, що використовує характеристики *A11* і *A12*, і т.д., доки не буде отриманий закінчений проект ПЗ, що відповідає всім специфікаціям.

Основний недолік зростаючого проектування, – це проблема інтеграції модулів одного рівня ієрархічної структури за допомогою модуля вищого рівня. Така інтеграція може бути надзвичайно складною, оскільки спроектовані окремо (і, можливо, різними програмістами) модулі нижчого рівня в загальному випадку мають неузгоджені інтерфейси і використовують різні способи подання даних. З цієї причини зростаюче проектування знаходить на практиці обмежене застосування, його використовують для створення програмних систем, що мають прототипи, структура яких добре відома і теоретично розроблена (ОС, мовні процесори і т.п.). У дійсності, при розробці ПЗ, спадне і зростаюче проектування не використовуються окремо, а взаємно доповнюють один одного. Наприклад, при зростаючому методі

зростаючому методі спочатку створюється «інструментарій» програмістів – загальні підпрограми введення-виведення, обслуговування структур даних, динамічного розподілу пам'яті і т.д. Подальше проектування ведеться спадним методом. З використанням спадного методу проектується склад модулів ПЗ, їхні функції і зв'язки щодо керування. Проектування зв'язків за інформацією на цій стадії носить ескізний характер. Детальна розробка міжмодульних інтерфейсів відбувається з використанням зростаючого методу. Чим нижчий рівень ієрархічної структури ПЗ, що займає модуль, тим частіше цей модуль застосовується, і, отже, зручність інтерфейсу модуля для його розробки значно впливає на ефективність усього ПЗ.

5.5.4 Вибір мови програмування

На етапі кодування модулів ПЗ велике значення має вибір мов програмування. На вибір мови програмування впливають:

- можливість створення довільних структур даних засобами мови;
- наявність конструкцій структурного програмування;
- перенесення програм з ЕОМ одного типу на ЕОМ іншого типу, з однієї ОС в іншу;
- зручність освоєння і використання, продуктивність кодування;
- ефективність закодованих програм.

В даний час для створення ПЗ САПР найбільшого поширення отримали алгоритмічні мови ФОРТРАН, ПЛ/1, ПАСКАЛЬ та Асемблер.

Алгоритмічна мова ФОРТРАН призначена тільки для науково-технічних розрахунків, дозволяє легко і швидко кодувати формули й ітераційні процеси над векторами і матрицями цілого і дійсного типів. Транслятори з мови ФОРТРАН є практично у всіх ОС і забезпечують високу ефективність об'єктного коду. Однак, примітивність цієї мови у відношенні типів і структур даних, відсутність динамічного розподілу пам'яті істотно обмежують її використання при розробці ПЗ САПР. Крім того, структурне програмування мовою ФОРТРАН можливо тільки з використанням спеціальних процесорів, що здійснюють переклад з «розширеної» мови ФОРТРАН, яка включає в себе конструкції структурного програмування, у стандартну мову ФОРТРАН.

Алгоритмічна мова ПЛ/1 має конструкції структурного програмування і багаті засоби для створення довільних структур даних. Але вона складна в освоєнні, її транслятори є в складі не всіх ОС, генерований нею об'єктний код поступається асемблерному у швидкодії і витратах ОП у 2÷3 рази.

Алгоритмічна мова ПАСКАЛЬ найчастіше використовується для

створення ПЗ САПР. Мова ПАСКАЛЬ значно простіша за мову ПЛ/1, хоча і має всі її можливості. У ній спеціально виключені конструкції, що призводять до неефективного об'єктного коду. Є великий досвід використання цієї мови для створення не тільки прикладних програм, але й ОС. Транслятори з мови ПАСКАЛЬ є на більшості ЕОМ. Сказане вище дозволяє рекомендувати цю мову як основну мову програмування ПЗ САПР.

Алгоритмічна мова Асемблер використовується при створенні складних програмних систем. Область застосування цієї мови в ПЗ САПР повинна бути обмежена тільки окремими модулями керувальної й обслуговувальних підсистем, оскільки його жорстка прив'язка до типу ЕОМ і ОС ускладнює адаптацію САПР до нових умов застосування. Можливо, що з поширенням трансляторів з мови СІ потреба в мові асемблера при розробці ПЗ САПР відпаде зовсім. Однак знання мови Асемблер, як і особливостей роботи використовуваної ЕОМ, є обов'язковим для програміста-професіонала, з якою б мовою він не працював.

Контрольні запитання

1. Структура програмного забезпечення (ПЗ) САПР.
2. Загальне ПЗ.
3. Призначення та основні функції операційних систем.
4. Структура спеціального програмного забезпечення САПР.
5. Характеристика моделювальних пакетів застосовуваних програм.
6. Обробка відеоінформації засобами графічних редакторів.
7. Обробка інформації засобами текстових редакторів.

6 ПАКЕТ ЛОГІЧНОГО МОДЕЛЮВАННЯ ACTIVE-HDL

6.1 Основні поняття VHDL

Для успішного моделювання в пакеті Active-VHDL необхідно добре розібратися з основними поняттями мови VHDL, такими як *архітектура* та *об'єкт*.

Архітектура – це тіло, яке пов'язано з оголошенням об'єкта і використовується для опису внутрішньої організації об'єкта проекту.

Тіло архітектури використовується для того, щоб описати поведінку, потік даних або структуру об'єкта проекту.

architecture *ім'я архітектури* **of** *ім'я об'єкта* **is**
оголошення архітектури
begin
паралельні інструкції
end [architecture] [*ім'я архітектури*];

Архітектура об'єкта описує внутрішні відносини між портами введення і виведення. Вона складається з двох частин: *оголошення* і *паралельних інструкцій*.

Об'єкт – це тіло, в якому описується зв'язок між проектом і його зовнішнім середовищем.

У ньому також визначаються оголошення і інструкції, які є частиною об'єкта проекту. Дане оголошення об'єкта може бути використане багатьма об'єктами проекту, кожен з яких має різну архітектуру. Так само, оголошення об'єкта є класом об'єктів, що мають однаковий інтерфейс.

entity is *ім'я об'єкта*
generic (список);
port (список портів);
end entity *ім'я об'єкта*;

Порт – це динамічний канал зв'язку між блоками і середовищем.

port (оголошення порту, оголошення порту, …);
– оголошення порту:
ім'я порту сигналу: **in** тип порту сигналу: = значення
ім'я порту сигналу: **out** тип порту сигналу: = значення
ім'я порту сигналу: **inout** тип порту сигналу: = значення
ім'я порту сигналу: **buffer** тип порту сигналу: = значення
ім'я порту сигналу: **linkage** тип порту сигналу: = значення

Порт є частиною блоку інтерфейсу. Є два типи портів: *зовнішній* і *внутрішній.*

Зовнішній порт – це порт, оголошений в об'єкті проекту.

Внутрішній порт – це порт, оголошений в блоці твердження.

Кожен елемент, що вноситься в список в портовому інтерфейсі, описує формальний порт, який надає канал для динамічного зв'язку між блоком і середовищем.

На практиці, порт найчастіше використовується в об'єкті і компоненті, де вони служать для оголошення сигналів об'єктів проекту або компонентів, відповідно.

У обох випадках, кожен елемент – це сигнал. Він може бути оголошений за допомогою слова signal. Після імені сигналу описується його режим. Режим описує напрям потоку даних через порт.

Є п'ять режимів для портів що використовуються в VHDL:

- *In – вхідний порт.* Змінні або сигнали можуть зчитувати значення з порту в режимі *in*, але не можуть присвоювати йому значення.
- *Out – вихідний порт.* Дозволяється зробити присвоювання значення сигналу порту режиму *out*, але сигнал не може прочитати з нього значення.
- *Inout – двонаправлений порт.* Може присвоювати значення сигналів і присвоювати своє значення сигналу.
- *Buffer – вихідний порт* з можливістю читання. Він відрізняється від двонаправленого порту тим, що він може бути модифікований тільки одним джерелом, тоді як двонаправлений порт змінюється багатьма джерелами.
- *Linkage.* Значення порту може бути прочитане або змінене, але тільки за допомогою появи відповідності об'єкта з режимом з'єднання.

Якщо порт оголошений із зарезервованим словом *bus*(шина), тоді сигнал оголошується як шина сигналів.

6.1.1 Типи даних мови VHDL

6.1.1.1 Тип Біт

Тип Біт **(Bit)** - розрядний тип, визначений у стандартному пакеті як обчислювальний тип даних, що має тільки два допустимих значення «0» і «1».

type bit is ('0', '1');

Тип Bit - основний тип для подання логічних значень.

6.1.1.2 Тип Розрядний вектор

Тип ***Розрядний вектор*** **(Bit Vector)** визначений в Стандартному пакеті як стандартний тип одномірного масиву з елементами розрядного типу.

type bit_vector is array (natural range <>) of bit;

Тип Bit_vector – необмежений вектор елементів розрядного типу. Розмір специфічного вектора визначається під час його декларації. Шлях індексації векторних елементів залежить від певного діапазону і може бути зростаючим чи спадним.

6.1.1.3 Логічний тип

Логічний тип **(Boolean)** визначений в стандартному пакеті як тип даних, що перелічується з двома можливими значеннями: помилковий і істинний.

type boolean is (false, true);

Тип boolean використовується для умовних операцій. Об'єкти boolean можуть використовуватися з будь-яким з операторів відношення <, >, =, = або /, =. Згідно з типом визначення, крайнє ліве значення типу Boolean помилкове, тому задане за замовчуванням значення будь-якого об'єкта типу Boolean помилкове.

6.1.1.4 Константа

Константа **(constant)**– об'єкт, значення якого не може бути змінене, якщо тільки це не визначено для проекту.

Константи можуть бути явно оголошені або вони можуть бути під елементами явно оголошених констант, або константами інтерфейсу. Константи, оголошені в пакетах можуть також бути константами часу виконання (затримки)

constant *constant_name*: type: = value;

Константа – об'єкт, значення якого ніколи не може бути змінене протягом процесу моделювання.

6.1.1.5 Std_Logic

Логічний тип Std_logic **–** це не частина VHDL стандарту. Він визначений в IEEE Std 1164.

```
type std_ulogic is (      'U',   -- Uninitialized
'X',  -- Forcing  Unknown
                  '0',   -- Forcing  0
                  '1',   -- Forcing  1
                  'Z',   -- High Impedance
                  'W',   -- Weak     Unknown
```

```
    'L',  -- Weak    0
    'H',  -- Weak    1
    '-'   -- Don't Care
    );
type std_ulogic_vector is array (natural range <> ) of std_ulogic;
function resolved (s:  std_ulogic_vector ) return std_ulogic;
subtype std_logic is resolved std_ulogic;
```

Тип Std_ulogic - розширення стандартного розрядного типу. Він визначає дев'ять значень, які дозволяють визначати логічні системи.

Щоб полегшувати специфікацію керованих множником сигналів (подібно шинам даних) пакет Std_Logic_1164 визначає функцію роздільної здатності для Std_ulogic, який, в свою чергу, служить як основа для оголошення типу Std_Logic.

6.1.1.6 Масив (Array)

Масив **(Array)** – тип, значення якого складається з елементів однакового підтипу (і, отже, того самого типу).

Елементи мають різні індекси (для одновимірного масиву) або послідовність індексів (для багатовимірного масиву). Кожний індекс повинен мати значення дискретного типу і повинен лежати в правильному індексному діапазоні.

type type_name is array (range) of element_type

type type_name is array (type range <>) of element_type

Массив – складний об'єкт, елементи якого мають той самий підтип. Кожен з елементів індексований одним або декількома індексами, потрібними для визначення дискретних типів. Номер індексу – номер виміру, тобто одновимірний масив має один індекс, двовимірний має два індекси і т д. Порядок індексів істотний і слідує за порядком вимірювань в оголошенні типу.

6.1.2 Оператори мови VHDL

6.1.2.1 Умовний оператор (If Statement)

Умовний оператор – оператор, який залежно від значення певних або більш відповідних умов вибирає для виконання один або не виконує жодного з включених в послідовність операторів .

if умова **then**
послідовні інструкції
end if;

if умова **then**
послідовні інструкції
else

послідовні інструкції
end if;

if умова **then**
послідовні інструкції
else if умова **then**
послідовні інструкції
else
послідовні інструкції
end if;

6.1.2.2 Оператор Циклу

Оператор циклу – це послідовність, яка включає певну кількість команд, що повинні бути виконані неодноразово.

мітка циклу: **while** умова **loop**
послідовні інструкції
end loop мітка циклу;

мітка циклу: **for** діапазон умов циклу **loop**
послідовні інструкції
end loop мітка циклу;

6.1.2.3 Оператор вибору

Оператор вибору – затвердження регістра, обраного для виконання однієї з декількох альтернативних послідовностей тверджень. Альтернатива вибору заснована на значенні пов'язаного виразу.

case вираз **is**
when вибір => послідовні інструкції
when вибір => послідовні інструкції
end case;

6.1.3 Бібліотека

library ім'я бібліотеки;

Оператор **Library** – визначає логічне ім'я для бібліотеки проекту, яка буде використовуватися в проекті. Бібліотека – це засіб зберігання заздалегідь проаналізованих модулів проекту. Практично це стосується головним чином пакетів.

Коли пакет повинен використовуватися в проекті, він повинен бути зроблений видимим проекту, щоб визначити, повинна використовуватися

заздалегідь проаналізованих модулів проекту. Практично це стосується, головним чином, пакетів.

Коли пакет повинен використовуватися в проекті, він повинен бути зроблений видимим проекту, щоб визначити, повинна використовуватися бібліотечна пропозиція (що робить бібліотеку видимою) чи ні, і зроблена пропозиція використання (що робить специфічні оголошення видимими).

Є дві визначені бібліотеки, які використовуються неявно в кожному проекті: STD і WORK. Перша з них містить стандартні пакети STANDARD і TEXTIO. Інша, робоча бібліотека, – містить всі розроблені і проаналізовані користувачем модулі проекту.

6.1.4 Редактор Блок-схем

Редактор Блок-схем – інструмент для графічного введення проектів VHDL. Якщо ваш проект VHDL має певну структуру, то ввести його опис графічно, як блок-схему, набагато швидше, ніж друкування сотень рядків вихідного тексту. Редактор Блок-схем перетворює діаграму автоматично в VHDL-код. У Active – HDL, можливо змішувати різні типи опису проекту.

Проект може складатися з декількох частин:

- VHDL-код;
- блок-схеми;
- діаграми кінцевих автоматів.

Для створення і редагування нового файла можна скористатися піктограмою **Add New File** або викликати **Майстер нового файла**.

На рисунку 6.1 показано приклад вигляду вікна редактора блок-схем.

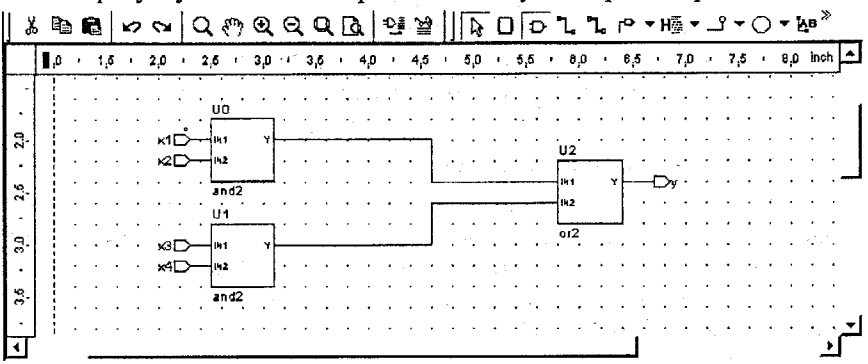


Рисунок 6.1 – Вигляд вікна редактора блок-схем

Кнопки панелі діаграми блок-схеми:

- Вводить режим вибору.
- Вводить режим рисування Fubs.
- Вводить режим рисування проводу.
- Вводить режим рисування шини.
- Вводить режим додання тексту.

Вводить режим додання термінала і додає термінал **in.**

Вводить режим додання термінала і додає термінал **out.**

Вводить режим додання термінала і додає термінал **inout.**

Вводить режим додання термінала і додає термінал **buffer.**

Вводить режим додання термінала і додає термінал шини **in.**

Вводить режим додання термінала і додає термінал шини **out.**

Вводить режим додання термінала і додає термінал шини **inout.**

Вводить режим додання термінала і додає термінал шини **buffer.**

Вводить режим додання VHDL інструкцій. Використовується для того, щоб додати або редагувати контекст.

Вводить режим додання VHDL інструкцій. Використовується для того, щоб додати або редагувати тіло архітектури.

Вводить режим додання VHDL інструкцій. Використовується для того, щоб додати сигнал.

Вводить режим додання VHDL інструкцій. Використовується для того, щоб додати універсальне оголошення.

Розміщує VCC символ.

Розміщує символ заземлення.

Розміщує глобальний провідниковий з'єднувач.

Розміщує глобальний шинний з'єднувач.

Відкриває або закриває набір елементів.

Вводить режим додання тексту.

Вводить режим рисування лінії.

Вводить режим рисування прямокутника.

Вводить режим рисування еліпса.

Вводить режим рисування дуги.

6.1.4.1 Елементи Блок-схеми

Кожен елемент на діаграмі має копію в згенерованому коді VHDL. Наприклад, термінали відповідають портам об'єкта, складові символи відповідають реалізації складових в тілі архітектури, і так далі. На рисунку 6.2 показано типову блок-схему, що містить елементи діаграми, які звичайно використовуються.

6.1.4.2 VHDL Інструкції

Користувач може додати до згенерованого коду додаткові інструкції, які не можуть бути графічно подані на блок-схемі. Це може бути досягнуто при доданні спеціальних текстових блоків з кодом VHDL. Є чотири типи таких блоків:

6.1.4.2.1 Контекст

Використовуючи цей блок, можна додавати додаткові входи до контексту. Контекст з'являється перед оголошенням об'єкта в згенерованому коді.

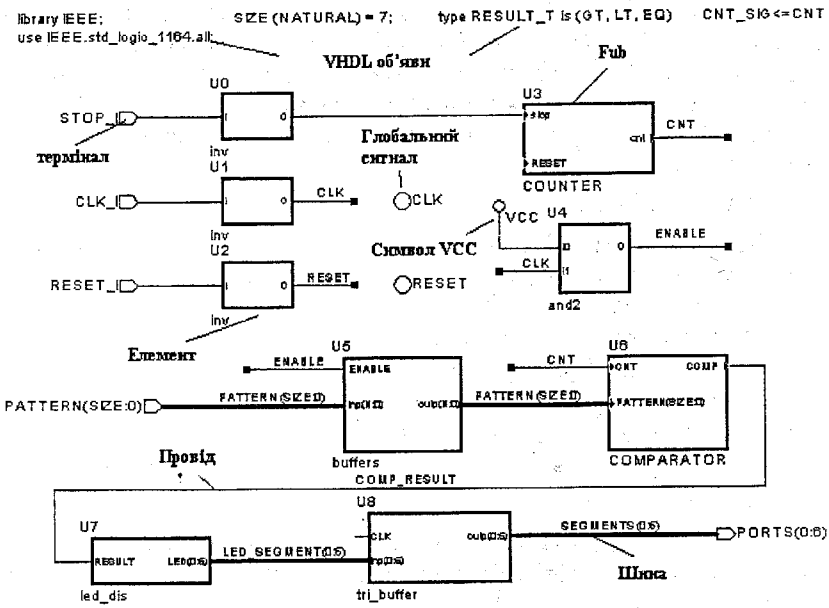


Рисунок 6.2 – Типові елементи діаграми блок-схеми

6.1.4.2.2 Паралельні Інструкції

Використовуючи цей блок, можна додавати додаткові паралельні інструкції до тіла архітектури, є діаграмою.

6.1.4.2.3 Оголошення

Використовуючи цей блок, можна додавати додаткові оголошення до декларативної частини тіла архітектури, є діаграмою.

6.1.4.2.4 Fubs

Fubs – символ, який створений і відредагований безпосередньо на листі діаграми. Безліч певних властивостей відрізняє fubs від регулярних символів. Можна вільно змінювати fubs символ, додавати і видаляти входи безпосередньо на схемі. Входи додаються автоматично кожний раз, коли підключається провід або шина до fubs.

Початкова мітка входу – ім'я проводу / шини. Входи видаляються автоматично кожний раз, коли роз'єднується провід або шина від fubs. Fubs – тільки в одному зразку. Це означає, що проект не може містити більше одного зразка даного fubs. З цієї причини, fubs не з'являється в

списку елементів. Fubs може бути створений тільки при використанні низхідної методології проекту. Іншими словами, fubs – символ, створений до визначення його змісту. Fubs може бути легко перетворений (конвертований). Подібно елементам, fubs - це графічне подання компонента в тілі архітектури. Порти такого компонента відповідають входам fubs. Входи помічені назвами відповідних портів. Опис fubs зберігається в бібліотеці проектів нарівні з об'єктом проекту, який його зображає. Ім'я fubs ідентичне імені об'єкта. Для недавно створеного fubs бібліотека містить відповідний вхід (званий "пустим" об'єктом і ідентифікований ім'ям fubs), але жоден відкомпільований модуль проекту не доступний.

6.1.4.2.5 Термінали

Термінал – графічний символ, який закінчує провід або шину. Можна призначати ім'я терміналу або залишати його не названим.

Є чотири види терміналів:

- **in** – відповідає порту **in;**
- **out** – відповідає порту **out;**
- **inout** – відповідає порту **inout;**
- **buffer** – відповідає порту **buffer.**

Вигляд порту **linkage** не підтримуваний редактором блок-схеми.

6.1.4.2.6 Символи

Символ – графічне подання компонента, що знаходиться в тілі архітектури. У проекті VHDL компоненти повинні бути пов'язані, щоб проектувати об'єкти.

Ім'я елемента ідентичне імені об'єкта, поданого цим об'єктом. Ім'я оголошеного компонента – завжди те саме, як і ім'я об'єкта. Порти відображені як входи на схемі елемента. Входи помічені іменами відповідних портів. Як тільки елемент був поміщений в діаграму, йому призначається унікальне ім'я зразка, яке використовується, щоб ідентифікувати його в межах листа діаграми.

6.1.4.2.7 Глобальні Сигнали

Глобальний сигнал – це мережа, яка неявно з'єднує всі не пов'язані входи в межах поточного листа блок-схеми, імена яких такі самі, як ім'я глобального сигналу. Щоб оголосити глобальний сигнал, необхідно розмістити глобальний сигнал на лист діаграми і призначати йому бажане ім'я.

6.1.4.2.8 Проводи

Проводи використовуються, щоб підключити символи і fubs на діаграмах. Кожний провід складає деяку дискретну мережу, тобто логічний зв'язок між елементами діаграми. Провід може мати ім'я або

бути не названим. Названі і не названі проводи керуються різними правилами з'єднання

Провід подається в VHDL скалярним сигналом. Ім'я сигналу – таке саме, як ім'я проводу. Якщо ім'я не визначене (неназваний провід), то редактор блок-схеми автоматично генерує ім'я. Тип сигналу може бути одним із стандартних скалярних типів або може бути визначений користувачем.

6.1.4.2.9 Шини

Шина – набір дискретних проводів, відображених як єдина товста лінія. Кожна шина зображає деяку мережу шин, тобто безліч логічних зв'язків між елементами діаграми. Подібно проводам, шини використовуються, щоб підключити елементи і fubs на діаграмах. Шині може бути призначено або не призначено мережеве ім'я. Крім того, для кожної шини повинен бути визначений індексний діапазон.

6.2 Моделювання в середовищі Active-VHDL

Active-VHDL містить простий у використанні **Майстер Проекту (New Design Wizard)**. Щоб створити новий проект, оберіть **New Design** з меню **Design**. Діалогове вікно дозволяє ввести ім'я проекту й ім'я каталогу, де будуть знаходитися файли проекту (рисунок 6.3).

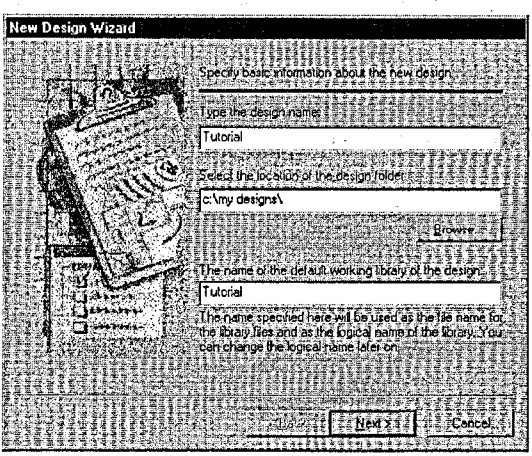


Рисунок 6.3 – Вікно **Design Wizard**

У поле **Type Design Name** введіть *Tutorial*, а також введіть чи оберіть бажане розташування папки проекту у відповідному полі. Натисніть кнопку **Next,** щоб перейти до наступної сторінки. Натисніть кнопку **Cancel**, щоб вийти з **Майстра Проекту (New Design Wizard)** без будь-яких змін.

6.2.1 Завдання ресурсів проекту

Наступне діалогове вікно (рисунок 6.4) дозволяє точно визначити ресурси створюваного проекту, в ньому можливо:

- Оперативно створити нові вихідні файли, що містять точно встановлені об'єкти.
- Додати існуючі файли з існуючих проектів.
- Імпортувати проект із Active-CAD.
- Створити чистий проект – жоден з компонентів не буде доданий до проекту.

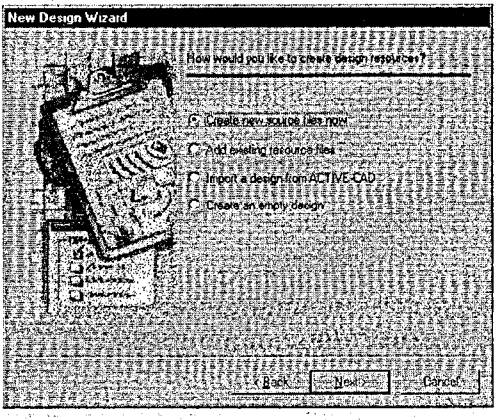


Рисунок 6.4 – Вікно ресурсів **Design Wizard**

Оберіть **Create new source files now**.

Натисніть кнопку **Back,** якщо необхідно заново ввести назву проекту і каталогу. Натисніть кнопку **Next,** щоб перейти до наступної сторінки. Натисніть кнопку **Cancel,** щоб вийти з режиму.

6.2.2 Створення макета вихідного файлу

У наступному діалоговому вікні можна додавати компоненти, які необхідно включити до проекту. Для цього необхідно натиснути кнопку **New** і потім ввести ім'я об'єкта (рисунок 6.5).

Стовпчик **Source Type** дозволяє вибрати текстовий опис VHDL чи графічний опис, використовуючи редактор **Кінцевих Автоматів** (Finite State Machine Editor).

Введіть *Counter*, як назву об'єкта, і виберіть VHDL-Code, як джерело. Виберіть пункт **Counter,** а потім натисніть на кнопку **Ports**. Після цього, використовуючи **Майстер Портів** (Port Wizard) введіть опис портів.

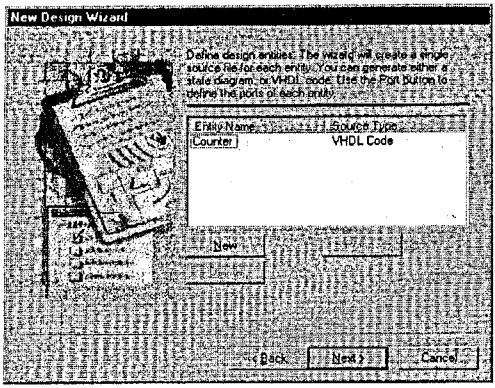


Рисунок 6.5 – Вікно об'єктів **Design Wizard**

6.2.2.1 Майстер Порта

Майстер Порта (Port Wizard) використовується для позначення портів (рисунок 6.6). Щоб ввести порти, натисніть на кнопку **New** і введіть ім'я. Ви можете вказати напрямок порту, використовуючи елемент **Direction**. Якщо необхідно ввести шину, то можна встановити її ширину, використовуючи елемент **Bus**.

Додайте три порти в такий спосіб:

- CLK – порт входу;
- RESET – порт входу;
- Q[3:0] – порт вихідна шина, діапазон [3:0].

Після того, як Ви додали порти, натисніть кнопку **Type.** На екрані з'явиться діалогове вікно, що дозволяє обрати тип порту.

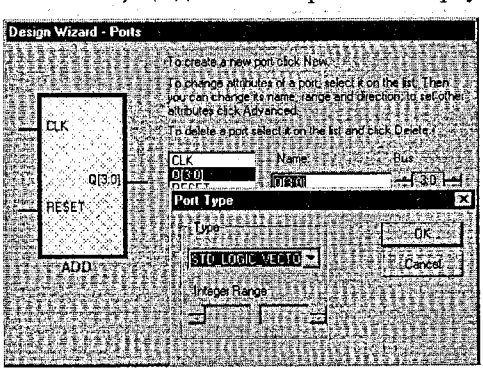


Рисунок 6.6 – Вигляд діалового вікна **Вікно Портів**

Оберіть опцію **STD_LOGIC** для одиничних портів і **STD_LOGIC_VECTOR** для шин. Це змусить згенерований шаблон включити бібліотеку IEEE і оголошення пакетів. Якщо натиснути **OK** у діалоговому вікні **Port Wizard'a**, то з'явиться діалогове вікно **Майстра**

Проектів (New Design Wizard). Натисніть на кнопку **Next**, щоб перейти до останньої сторінки.

6.2.2.2 Підтвердження реквізитів проекту

Останнє діалогове вікно – вікно підтвердження. Усі реквізити проекту показані у відповідному вікні. Якщо вони записані правильно, натисніть кнопку **Finish**.

Використовуйте кнопку **Back**, якщо установки введені неправильно і кнопку **Cancel**, якщо необхідно відмовитися від створення нового проекту.

6.2.3 Вікно перегляду Проекту

Вікном **перегляду Проекту** називається вікно, що показує вміст проекту. У результаті попередніх операцій отримаємо таке вікно (рисунок 6.7):

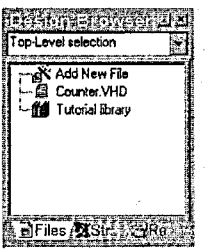


Рисунок 6.7 – Вигляд вікна **перегляду Проекту**

Виберіть файл *Counter.vhd* у вікні **перегляду Проекту**, щоб ввести назву вихідного файлу. Виберіть **Compile** з меню **Design** і спостерігайте зміни у **Вікні перегляду Проекту**. Позначка біля файлу *Counter.vhd* змінює блакитний кольор на зелений.

Шоб розширити перегляд необхідно натиснути знак "+" ліворуч від файла.

Вікно показує пари архітектура-об'єкт. Двічі натисніть позначення *Counter.vhd*, щоб викликати **HDL Редактор** (рисунок 6.8):

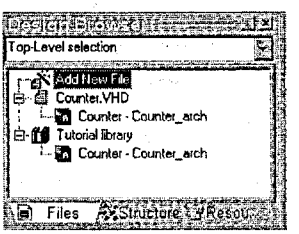


Рисунок 6.8 – Вигляд вікна **перегляду Проекту** зі структурою об'єктів

6.2.4 Редагування Коду. HDL Редактор

HDL Редактор – це текстовий редактор з **VHDL** ключовими словами, зафарбованими певним кольором і стандартними засобами редагування.

Якщо всі попередні кроки зроблені правильно, то на екрані будете вікно, вигляд якого наведений на рисунку 6.9. Згенерований код – шаблон, заснований на попередньо зроблених установках портів.

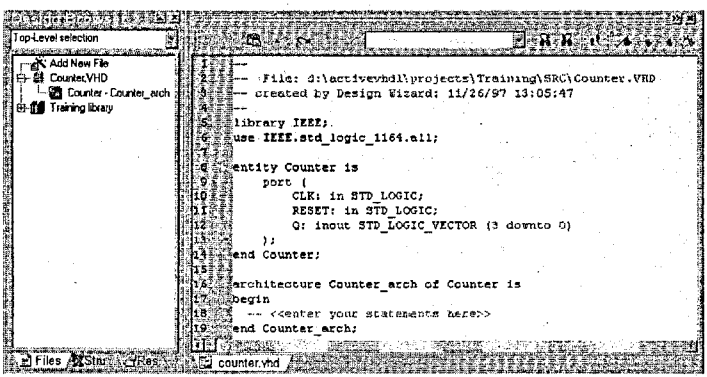


Рисунок 6.9 – Вигляд вікна **HDL Редактора**

6.2.4.1 Мовний Помічник

Для приклада опишемо структуру **"Лічильника"**.

Якщо відмітка **Language Assistant** у меню **Tools** не поставлена, то поставте її зараз. Якщо розгорнути дерево **Tutorial,** з'являться внутрішні пункти і повинно відображатися вікно, показане на рисунку 6.10.

Після натискання на **Counter** у вікні попереднього перегляду, що з'являється, є опис мовою VHDL лічильника двійково-десяткового коду. Далі необхідно вставити код опису функціонування обраного пристрою у шаблон. Для цього:

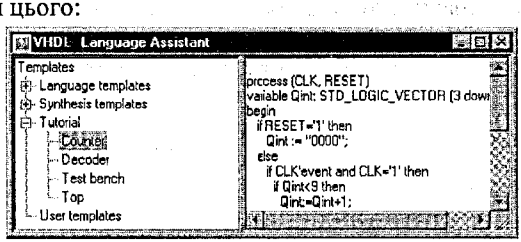


Рисунок 6.10 – Вигляд вікна **Language Assistant**

1. У вікні **VHDL редактора** необхідно знайти **begin** у коді *Counter.vhd.*

2. Встановити курсор у наступному рядку після **begin** у блоці архітектури.
3. Вибрати **Counter** у позначенні дерева **Language Assistant**.
4. Обрати **Use** зі скороченого меню.
5. Шаблон буде негайно поміщений в обране місце на кроці 2.

6.2.4.2 Додання бібліотек

Відредагований код вимагає, щоб були підключені деякі додаткові пакети. Щоб це зробити після *use IEEE.std_logic_1164.all* необхідно додати такий рядок: **use IEEE.std_logic_unsigned.all;**

Результат цієї дії показано на рисунку 6.11.

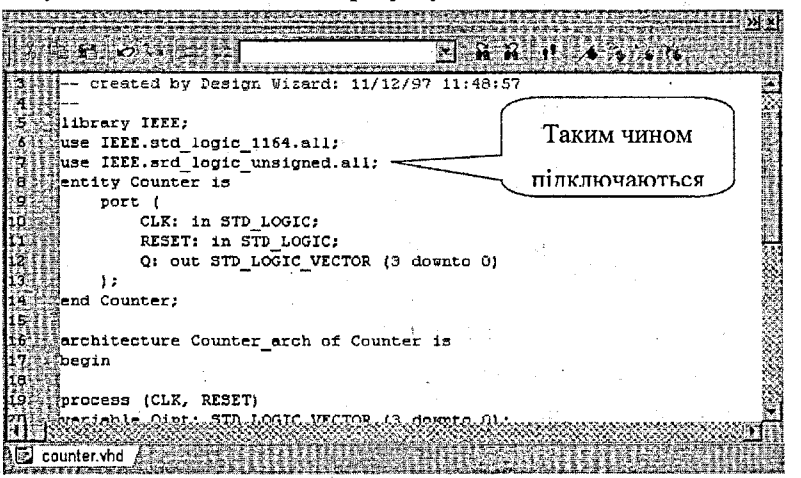


Рисунок 6.11 – Вікно **HDL Редактора** з кодом лічильника

Таким чином редагуються пристрої проекту.

6.2.4.3 Контроль Синтаксису

Для перевірки правильності написання коду необхідно перейти до вікна **Перегляду проекту** (Design Browse), для чого виберіть *Counter.vhd* і натисніть праву кнопку миші. Виберіть опцію **Compile** зі скороченого меню (рисунок 6.12).

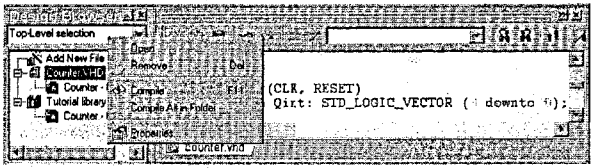


Рисунок 6.12 – Проведення перевірки синтаксису коду

Якщо є будь-яка помилка, то іконка стане жовтою, помилкові рядки підкреслені, і вікно VHDL консолі буде містити опис помилки. Якщо необхідно побачити, як вони будуть відображені, видаліть перший символ з бібліотечного запису *Counter* і виконайте трансляцію (рисунок 6.13).

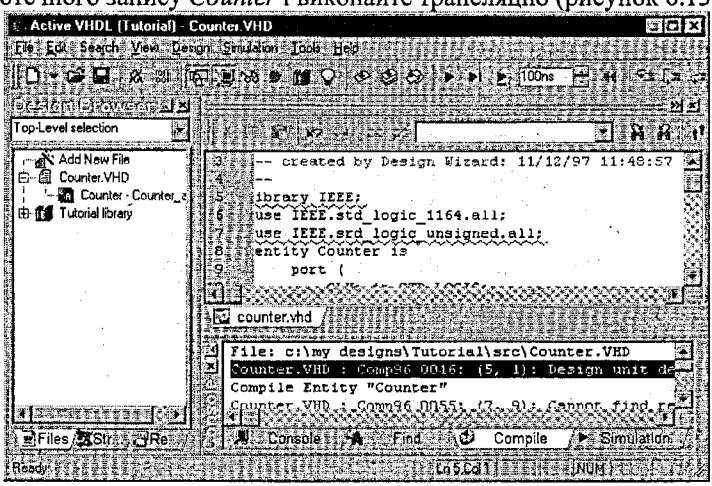
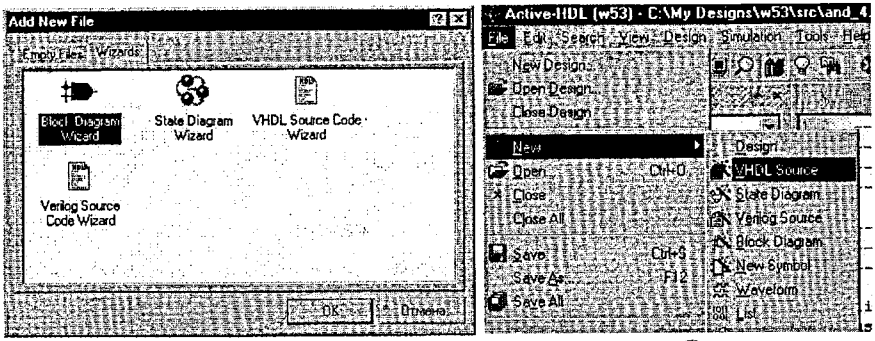


Рисунок 6.13 – Вікно **HDL Редактора** з індикацією помилок

Щоб повернутися до попереднього вікна, вільного від опису помилок, необхідно ввести символ, що був видалений.

6.2.4.4 Додання Файлів до Проекту

Щоб додати файл, необхідно двічі натиснути позначення **Add New File** у дереві **Вікна перегляду Проекту**. З'явиться наступне діалогове вікно, що допоможе вибрати спосіб, яким буде доданий до проекту новий файл (рисунок 6.14,а). Аналогічну дію можна виконати, якщо виконати команду **File → New → VHDL Sourse** (рисунок 6.14,б).



а) б)

Рисунок 6.14 – Команди меню **New Vhdl Source**

Після виконання цієї дії необхідно повторити дії, починаючи з пункту **"Створення макета вихідного файлу"**, для наступного пристрою.

6.2.5 Створення основного файлу проекту

Необхідно створити новий чистий **VHDL** файл одним із способів, які описані вище. Далі необхідно перейти до **Language Assistant** і знайти позначення **Top**. Обираємо **Use** зі скороченого меню, щоб вставити шаблон у код. Переходимо до меню **File** і обираємо опцію **Save As**, зберігши файл як *Top.vhd*. Вміст **HDL-редактора** після цих дій наведено на рисунку 6.15.

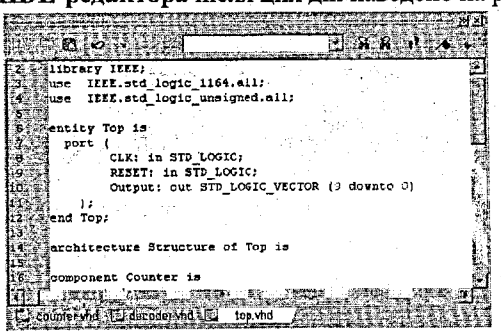


Рисунок 6.15 – Вміст **HDL-редактора**

Тепер необхідно перекомпілювати весь проект, використовуючи опцію **Compile All,** доступну для правої кнопки висхідного меню.

6.2.6 Перегляд структури проекту

Для точного визначення об'єкта верхнього рівня натисніть кнопку висхідного списку, щоб побачити доступні пари архітектури-об'єкти. Виберіть позначення "top-structure" (рисунок 6.16).

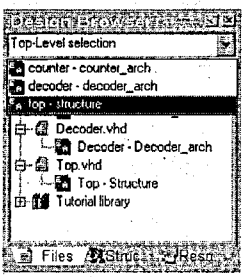


Рисунок 6.16 – Вибір високорівневої архітектури

Те саме можна зробити, обираючи позначення "top-structure" у дереві і вибираючи опцію **Set As Top level**, доступну у висхідному меню, що викликається правою кнопкою миші. Також можна використовувати команду **Settings**, доступну в меню **Design**. Це покаже діалогове вікно **Установок Проекту**, у якому Ви зможете вибирати **об'єкт верхнього рівня**. У **Вікні перегляду Проекту** виберіть вкладку **Structure** (рисунок 6.17).

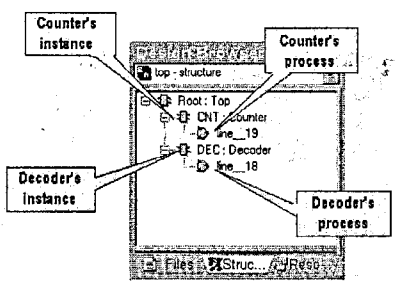


Рисунок 6.17 – Компоненти вкладки **Структура**

Вкладка Структура показує структуру проекту. У проекті **Top**, що має атрибут **Root**, є дві складові: *Лічильник* і *Дешифратор*. Кожна з цих складових виконує один процес, що показано в дереві. Також показані усі пакети, що використовуються.

6.2.6.1 Перегляд Локальних Даних

Кожен проектований пристрій може містити порти, сигнали і змінні. **Active-VHDL** допускає простий перегляд через дані пристрої. Якщо натиснути складові, позначені на вкладці **Структура** вікна **перегляду Проекту**, то локальні дані будуть показані в підвікні (рисунок 6.18).

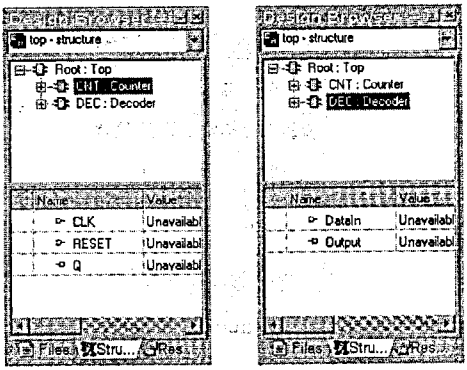


Рисунок 6.18 – Вибір моделі портів

6.2.7 Моделювання створеної схеми

Щоб почати моделювання, необхідно спочатку ініціалізувати стимулятор, використовуючи опцію **Initialize Simulation** з меню **Simulation.** Після того, як моделювальний пристрій було ініціалізовано, відкрити нове вікно **Форми сигналу** (Waveform Window) і натиснути кнопку панелі **New Waveform**. З'явиться нове вікно форми сигналу (рисунок 6.19).

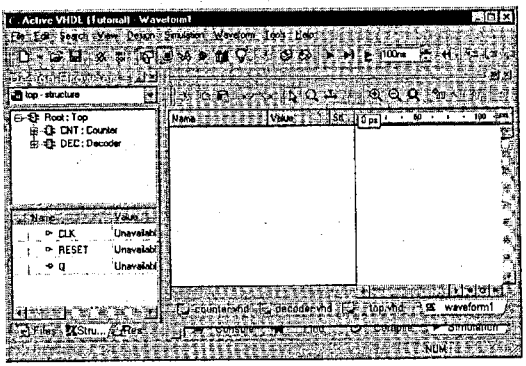


Рисунок 6.19 – Додання сигналу з **Вікна перегляду Проекту**

Сигнали для моделювання можуть бути додані переміщенням їх у вікно. Для цього на вкладці **Structure** вікна **Перегляду Проекту** необхідно вибрати складову потрібних портів і просто натиснувши ліву кнопку миші, перемістити усі разом до лівої частини вікна **Форми сигналу**, після чого відпустити кнопку. Це типова процедура "drag-and-drop"–взяв і перемістив (рисунок 6.20).

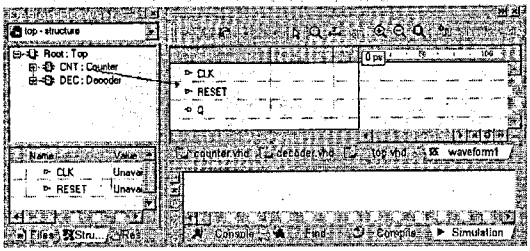


Рисунок 6.20 – Додання сигналу з **Вікна перегляду Проекту**

Якщо необхідно видалити сигнал, виберіть його і натисніть праву кнопку. З'явиться висхідне меню, що містить пункт **Delete.** Виберіть цю опцію, щоб видалити сигнал.

6.2.7.1 Трасування вихідного тексту

Active-VHDL дозволяє рухатися по вихідному тексту під час моделювання. Вікно з текстом автоматично відкривається після натискання на кнопку моделювання. Рядок, що буде виконаний наступним, відмічений в **HDL-редакторі** жовтим кольором.

Натисніть на одну з кнопок: **Watch** і **Process,** щоб переглянути додаткові вікна **Watch** і **Process**.

Вікно **Process** показує статус процесів проекту. Вікно **Watch** дозволяє перевіряти значення змінних і сигналів.

6.2.7.2 Встановлення стимуляторів

У лівій частині вікна **Перегляду сигналів** виберіть сигнал CLK. Натисніть праву кнопку, щоб викликати висхідне меню (рисунок 6.21).

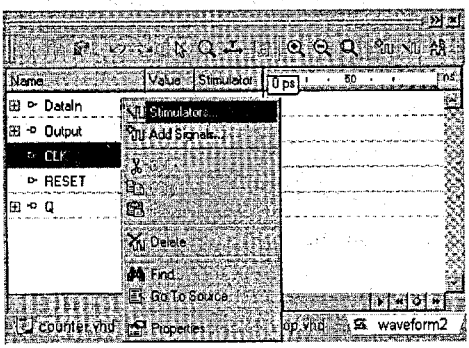


Рисунок 6.21 – Вікно встановлення **Стимуляторів** сигналу

Необхідно вибрати пункт **Stimulators,** а потім пункт **Clock** (рисунок 6.22).

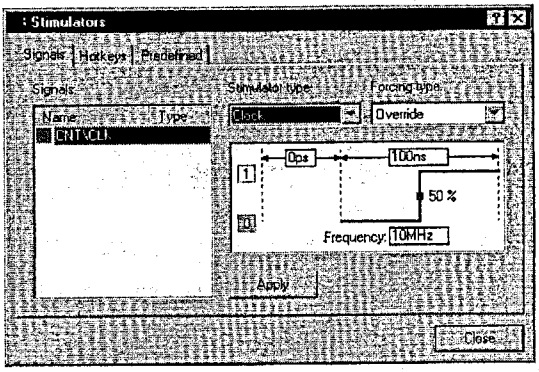


Рисунок 6.22 – Вікно встановлення частоти генератора для сигналу

Встановіть покажчик миші в поле **Frequency** і оберіть необхідне значення, наприклад, 100 MHz. Натисніть кнопку **Apply,** щоб запустити моделювання.

Далі необхідно задати значення сигналу *RESET* у вікні перегляду сигналів діалогу **Stimulators** з висхідного меню, для чого оберіть пункт **Formula** зі сценарію вибору. Коли з'явиться діалогове вікно, наберіть вираз:

1 0, 0 10000 (рисунок 6.23).

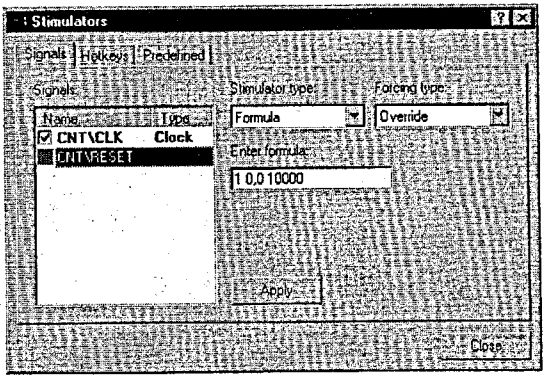


Рисунок 6.23 – Задання форми сигналу у вигляді формули

Натисніть на **Apply,** потім на **Close.**

6.2.7.3 Початок моделювання

При моделюванні можливо або зробити один крок моделювання, що корисно для налагодження вихідного коду програми, або тривале моделювання, для швидкого аналізу проекту і порівняння результатів (рисунок 6.24).

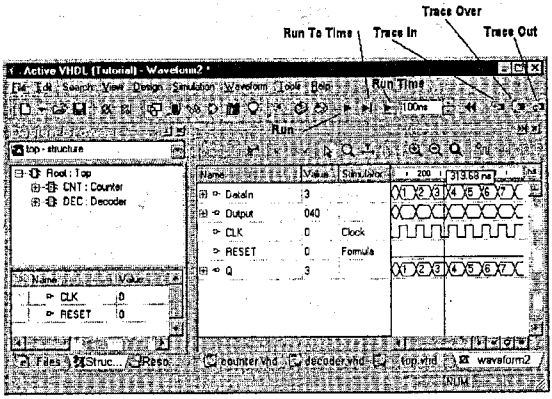


Рисунок 6.24 – Вибір типу процесу моделювання

Для виконання окремого кроку моделювання натисніть на кнопку **Trace Over**. Після натискання на кнопку **Run For** будуть отримані результати на вкладці **Форми_сигналу_2**, як наведено на рисунку 6.25.

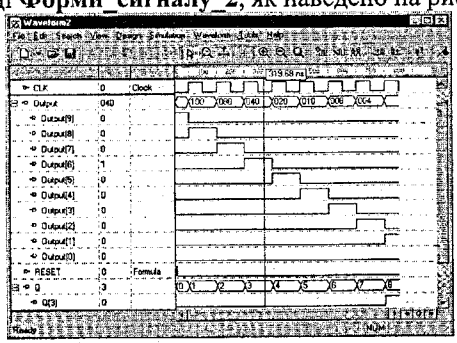


Рисунок 6.25 – Відображення результат моделювання

Вихідна шина розширюється натисканням на знак "+" поруч з позначенням. Можна натискати, наприклад, на закладку **Design**. При цьому у вікні **HDL Редактора** відобразиться код, що підлагоджується (рисунок 6.26).

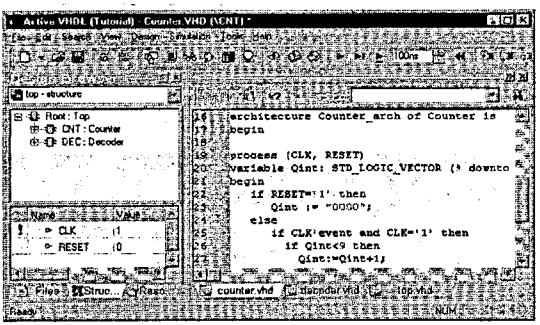


Рисунок 6.26 – Трасування коду у вікні **Редактора HDL**

Завершується моделювання вибором пункту **End Simulation** у меню **Simulation**.

6.2.7.4 Додання Випробувального стенда

Щоб додати випробувальний стенд, що є необхідним навколишнім середовищем для випробовування проекту, необхідно виконати такі дії:

- Створіть новий файл, використовуючи ту саму процедуру, що і для *Top.vhd*.
- Вставте шаблон, який має назву **Випробувальний стенд** від **Помічника Мови.**
- Тепер **HDL Редактор** містить **Test_Bench** опис.
- Натисніть кнопку панелі **Compile All**, активувавши її правою клавішею миші.
- Виберіть об'єкт **Test_Bench** як об'єкт верхнього рівня.

Після цих кроків вікно **Редактора HDL** буде мати вигляд, який наведено на рисунку 6.27.

6.2.7.5 Запуск Випробовувача

Випробовувач – це тестове середовище для випробовувань. Він включає *Stimuli Generator*, *Unit Under Test* і *Verifier,* для проведення всіх циклів моделювання всередині себе. Для створення випробовувача спочатку необхідно видалити попередні сигнали з вікна сигналів (випробовувач буде створювати сигнали **Clock** і **Reset** самостійно). Додайте всі кореневі сигнали. Потім натисніть на кнопку **Run Time** і виберіть час моделювання, як показано на рисунку 6.28.

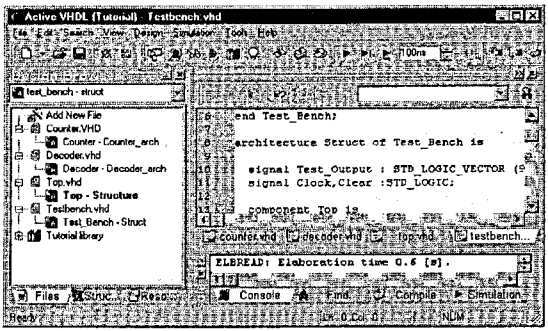


Рисунок 6.27 – Структура випробувального стенда у вікні **Редактора HDL**

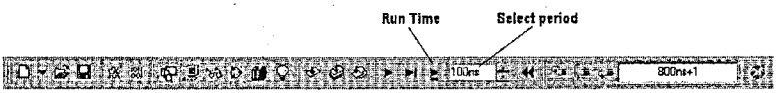


Рисунок 6.28 – Інструментальний рядок кнопок для моделювання

Результати моделювання показані у вікні **Перегляду Сигналів** (рисунок 6.29).

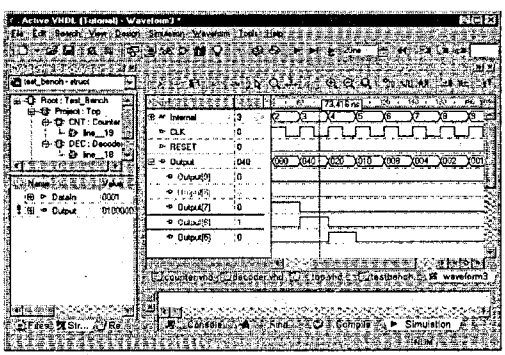


Рисунок 6.29 – Результати моделювання

6.2.8 Моделювання з використанням ієрархії

Метод ієрархічного проектування корисний при проектуванні складних схем, які складаються з багатьох елементів, визначаючи модулі для окремих частин схеми. Модулі далі розглядаються як елементи. Ієрархічні структури використовуються при зображенні принципових схем, які складаються з однотипних модулів.

Розглянемо приклад побудови схем з ієрархією на прикладі чотирирозрядного суматора. Створюємо новий пустий проект (**Create an empty design**). Для побудови чотирирозрядного суматора необхідно

спочатку побудувати однорозрядний суматор з урахуванням переносу, а потім на його базі побудувати чотирирозрядний суматор з урахуванням переносу.

Зробимо такі позначення:

A – розряд першого числа;

B – розряд другого числа;

P0 – перенос з попереднього розряду;

S – результат підсумовування;

P – перенос в наступний розряд.

Таблиця істинності функціонування пристрою буде мати вигляд (таблиця 6.1).

Таблиця 6.1 – Таблиця істинності функціонування пристрою

A	B	P0	S	P
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Для побудови однорозрядного суматора з урахуванням переносу необхідні такі логічні елементи три АБО, два І та виключне АБО. Візьмемо ці елементи зі стандартного набору елементів бібліотеки **altera_exemplar** або з бібліотеки проекту, якщо базові елементи були створені користувачем.

Для того, щоб додати ці бібліотеки необхідно:

- дати команду **File → New → Block Diagram** основного меню (або, двічі клацнувши лівою кнопкою мишки на команді **Add New File** у вікні **Design Browser**, обрати **Block Diagram**);
- задати ім'я проектованої схеми;
- відкрити вікно **Symbols Toolbox**, клацнувши кнопкою **Show Symbols Toolbox** на панелі інструментів;
- у вікні **Symbols_Toolbox** натиснути праву клавішу мишки і у вікні висхідного меню вибрати пункт **Choose_Libraries...**, як це показано на рисунку 6.30.
- вибрати потрібну бібліотеку;
- у вікні **Symbols_Toolbox** після розкриття з'являться елементи обраної бібліотеки.

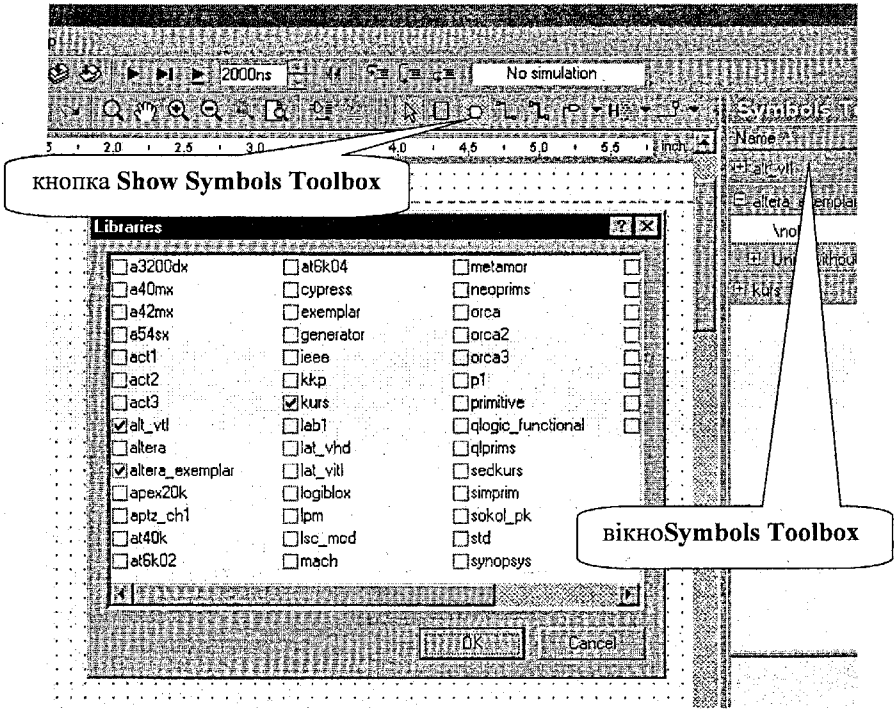


Рисунок 6.30 – Додання бібліотеки

Помістити потрібний елемент в потрібне місце можна перетягуванням його мишкою з вікна **Symbols_Toolbox** у вікно редактора блок-схеми. З'єднавши елементи і додавши термінали входів та виходів, можна починати моделювання.

Після всіх вище вказаних дій потрібно згенерувати VHDL-код та відкомпілювати його.

Отримана схема має вигляд, як наведено на рисунку 6.31.

Далі компілюється ця схема та і будуються часові діаграми. Впевнившись, що схема працює вірно, переходимо до наступного етапу (рисунок 6.32).

В цьому самому проекті створюємо новий файл блок-діаграм. І у ньому підключаємо як бібліотеку створений нами файл з однорозрядним суматором з урахуванням переносу. Створений нами суматор буде мати вигляд одного блоку з усіма входами та виходами.

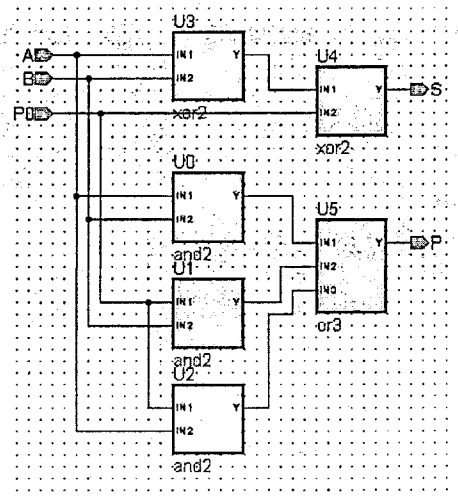


Рисунок 6.31 – Принципова схема однорозрядного суматора з переносом

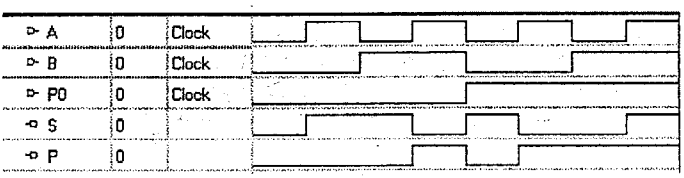


Рисунок 6.32 – Часові діаграми роботи однорозрядного суматора з переносом

Побудуємо схему чотирирозрядного суматора, використовуючи даний елемент. Схема буде мати вигляд, як наведено на рисунку 6.33.

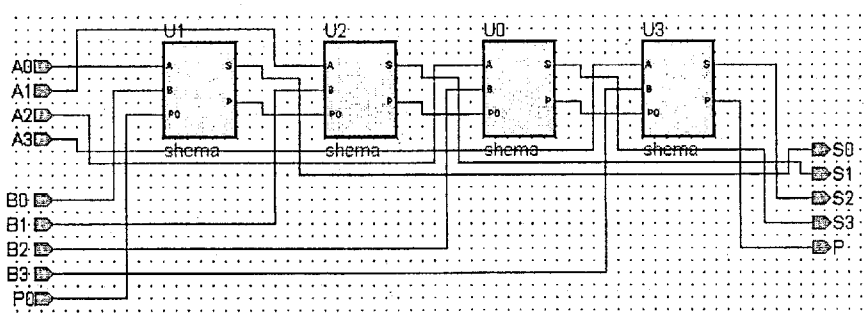


Рисунок 6.33 – Принципова схема чотирирозрядного суматора з переносом

Далі, після компіляції проекту, отримуємо часові діаграми для

чотирьох контрольних наборів, які наведено на рисунку 6.34.

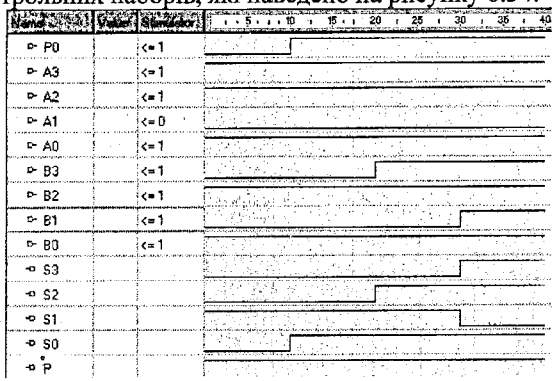


Рисунок 6.34 – Часові діаграми роботи чотирирозрядного суматора з переносом

6.2.9 Дослідження запам'ятовувальних пристроїв

В цьому розділі розглядається технологія логічного моделювання *оперативного запам'ятовувального пристрою* (ОЗП) та *постійного запам'ятовувального пристрою* (ПЗП) в середовищі Active-HDL.

Оперативний запам'ятовувальний пристрій має:

- 8-розрядний вхід *DATA_IN* (7 **downto** 0) типу *STD_LOGIC_VECTOR* для передачі вхідних даних;
- 8-розрядний вхід *ADDRESS* (7 **downto** 0) типу *STD_LOGIC_VECTOR* для визначення адреси комірки, з якою будуть виконуватись операції (тим самим визначається розмір ОЗП 8×256=2048Байтів);
- вхід синхронізації *CLK*;
- вхід дозволу на записування *WE*;
- вхід дозволу зчитування *RE* (типу *STD_LOGIC*);
- 8-розрядний вихід *DATA_OUT* (7 **downto** 0) для виведення даних (типу *STD_LOGIC_VECTOR*).

Робота ОЗП має здійснюватися у таким спосіб:

1. У стані збереження байта на виході ОЗП постійно утримується високий імпеданс ("ZZZZZZZZ").

2. Якщо *WE* = '0' і *RE* = '1', то на вихід ОЗП подається значення байта, що знаходиться за адресою *ADDRESS*.

3. Якщо *WE* = '1' і *RE* = '0', то в ОЗП записується байт *DATA_IN* за адресою *ADDRESS*.

4. Всі інші комбінації *WE* та *RE* розглядаються як стан збереження байта.

5. Робота ОЗП має бути синхронізована за сигналом *CLK*.

Постійний запам'ятовувальний пристрій має:

- 8-розрядний вхід *ADDRESS* (7 **downto** 0) типу *STD_LOGIC_VECTOR* для визначення адреси комірки, з якої будуть вибиратися дані;
- вхід синхронізації *CLK*;
- вхід дозволу зчитування *RE* (типу *STD_LOGIC*);
- 8-розрядний вихід *DATA_OUT* (7 **downto** 0) для вивединня даних (типу *STD_LOGIC_VECTOR*).

Таким чином визначається об'єм ПЗП 8×256=2048Байтів=2Кбайти.

Робота ПЗП має здійснюватися таким чином:

1. У стані збереження байта на виході ПЗП постійно утримується високий імпеданс ("ZZZZZZZZ").
2. Якщо *RE* = '1', то на вихід ПЗП подається значення байта, що знаходиться за адресою *ADDRESS*.
3. Якщо *RE* = '0', то це розглядається як стан збереження байта.
4. Робота ОЗП має бути синхронізована за сигналом *CLK*.

Контрольні запитання

1. Характеристика моделювального середовища VHDL.
2. Використання мови VHDL для структурного моделювання. Створення базових елементів.
3. Використання мови VHDL для структурного моделювання. Побудова образу принципової схеми.
4. Алфавіт та синтаксис мови VHDL.
5. Опис принципових цифрових схем засобами мови VHDL.
6. Використання мови VHDL для структурного моделювання. Використання ієрархії.
7. Використання мови VHDL для структурного моделювання. Дослідження запам'ятовувальних пристроїв.
8. Принципи організації процесу моделювання в середовищі VHDL.

7 МОДЕЛЮВАННЯ В СЕРЕДОВИЩІ PCAD

7.1 Загальні відомості про систему проектування PCAD

Система автоматизованого проектування радіоелектронної апаратури PCAD працює під управлінням операційної системи MS-DOS версії 3.0 і вище. Мінімальний об'єм оперативної пам'яті для монопольного користування системою повинен бути не менше 590 Кбайтів без розширеної пам'яті і 550...570 Кбайтів з розширеною пам'яттю, об'єм пам'яті на жорсткому магнітному диску не менше 10 Мбайтів, необхідний сопроцесор (в окремих випадках можна робити і без нього).

Програмний комплекс системи PCAD включає в себе взаємопов'язані пакети програм та окремі програми, які утворюють систему наскрізного проектування радіоелектронної апаратури.

Структурна схема підсистеми моделювання цифрових пристроїв на рисунку 7.1. На лініях зв'язку на цьому рисунку вказані стандартні розширення імен файлів, за допомогою яких відбувається обмін інформацією між окремими модулями.

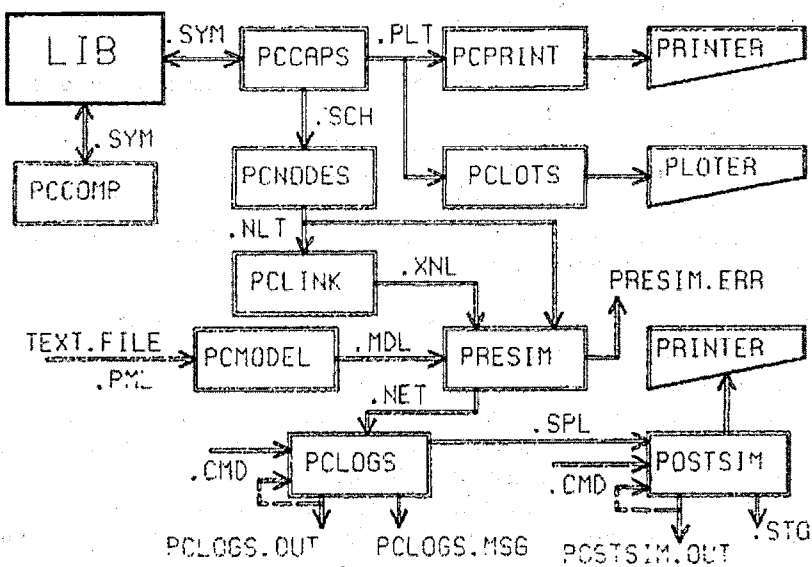


Рисунок 7.1 – Структура пакета PCAD

Підсистеми моделювання складаються із таких програмних модулів:

PCCAPS – графічне введення і редагування символів електронних компонентів (створення файлів з розширенням *.SYM) та принципових електричних схем (створення файлів з розширенням *.SCH).

PCCOMP – редагування інформації про контакти компонентів, упакування декількох секцій у корпус інтегральної мікросхеми та розміщення контактів на графічному зображенні компонента у файлах з розширенням *.SYM.

PCMODES – викликання списку електричних зв'язків із графічного образу принципової схеми у вигляді двійкового файла, який має розширення *.N1T.

PCLINK – об'єднання взаємопов'язаних списків електричних зв'язків схем, які складаються із декількох сторінок або які мають ієрархічну структуру, в єдиний список у файлі з розширенням *.XNL.

PCPRINT – виведення креслення електричної принципової схеми на принтер.

PROBE – графічне відображення на екрані дисплея документування результатів моделювання принципових схем.

PCMODEL – перетворення текстового файла з описом логічної моделі нестандартного компонента цифрового пристрою, створеного користувачем мовою функціонально-логічного моделювання PML, у формат програми PRESIM (утворюються файли з розширенням *.MDL).

PRESIM – складання логічного опису цифрового пристрою у форматі програми моделювання PGLOGS (створюються файли з розширенням *.NET).

PCLOGS – логічне асинхронне трійкове моделювання цифрових пристроїв.

POSTSIM – графічне відображення на екрані дисплея і документування результатів логічного моделювання цифрових пристроїв за допомогою програми PCLOGS.

Крім того, підсистеми включають не відмічені на рисунку 7.1 допоміжні програми (утиліти).

7.2 Загальні принципи роботи з графічним редактором

Графічний редактор принципових електричних схем PCCAPS використовується для розв'язання двох задач:

1. Побудова (редагування) графічного зображення (символа) окремого елемента (компонента) принципової електричної схеми (створюється файл з розширенням *.SYM);

2. Побудова (редагування) принципової електричної схеми аналогового або цифрового пристрою (створюється файл з розширенням *.SCH).

Програма PCCAPS викликається безпосередньо за допомогою командного рядка:

> pccaps або *>pccaps -r*

Після виклику програми PCCAPS (без зазначення параметра *-r* в командному рядку) на екрані з'являється початкове меню програми.

Configure PC-CAPS (встановлення конфігурації).
»Edit data base« (редагування бази даних).
Exit PC-CAPS (вихід із програми).

Один із цих рядків яскраво підсвічується і виділяється дужками >> <<. Курсор, що світиться, переміщується на інший рядок натисненням клавіші «пропуск» або за допомогою маніпулятора «мишка». Для підтвердження вибраного режиму достатньо натиснути клавішу <**Enter**>.

Після вибору у початковому меню режиму *Edit database* екран дисплея форматується і розбивається на кілька зон. *Зона побудови креслення* розмічається координатною сіткою. *Зона меню команд* призначена для команд графічного редактора, розташованих в двох колонках. Вибрана команда (вона називається активною) поміщається у прямокутну рамку. Рядок повідомлень призначений для діалогового зв'язку між користувачем і програмою. В рядку станів виводиться інформація про поточні параметри активної команди.

Побудова креслення схеми виконується за допомогою маніпулятора «мишка», який переміщується по горизонтальній поверхні робочого столу. При цьому на екрані дисплея синхронно переміщується курсор у вигляді хреста. Маніпулятор "мишка" має дві або три кнопки. Ліва кнопка (кнопка 1) призначена для вибору команди або місцеположення курсора на екрані (термін "вибір" означає одноразове натиснення кнопки 1). Права кнопка (кнопка 2) служить для відміни команди або попереднього режиму вибору в складних командах. Якщо маніпулятор "мишка" відсутній, то положення курсора встановлюється за допомогою функціональних клавіш. Функціональна ліва кнопка маніпулятора "мишка" відповідає клавіші "**Пропуск**", а права-клавіша "**ESC**".

В програмі PCCAPS повна інформація про креслення заноситься у 18 шарів. На кожній фазі роботи з графічним редактором не вся інформація є необхідною, тому частину шарів роблять невидимими, щоб не перевантажувати креслення.

Шари можуть бути зафарбовані в будь-який з 15-ти кольорів. Кожен шар може мати один із трьох станів (статусів):

OFF – шар невидимий і недоступний для редагування;

ON – шар видимий, але недоступний для редагування;

ABL – шар видимий і може стати активним (доступним для редагування) в режимі *ABL A*.

Команди рядка станів використовується для визначення параметрів (крок сітки, товщина лінії тощо).

Структура командного рядка залежить від того, яка команда активізована в меню команд.

Для вибору режимів роботи графічного редактора є дві команди **SYMB** і **DETL**. Команда **SYMB** встановлює режим введення (редагування) графічних зображень компонентів (символів) принципової схеми, при

цьому меню команд зафарбовано в зелений колір. Команда **DETL** визначає режим введення/редагування креслення принципової електричної схеми; при цьому меню команд зафарбовано в зелений колір.

Зона меню команд вміщує перелік основних команд і підкоманд, щоб вибрати команду з цього меню необхідно помітити курсором ім'я команди та натиснути кнопку 1 маніпулятора "мишка" або клавішу "пропуск" на клавіатурі. Якщо у вибраній команді є підкоманди, вони підсвічуються жовтим кольором в середній частині лівої колонки зони меню команд. Після вибору підкоманди вона активізується, в рядку стшів з'являються параметри, а в рядку повідомлень – інформація про подальші дії.

Перелік команд графічного редактора PCCAPS в черзі їх розташування в зоні меню команд наведено в [11].

7.3 Моделювання цифрових пристроїв

Перед моделюванням цифрового пристрою за допомогою програми PCLOGS необхідно виконати таке (див. рисунок 7.1). Спочатку за допомогою графічного редактора PCCAPS складається принципова схема пристрою і заноситься в файл з розширенням *.SGH. Потім із цього файла за допомогою програми PCNODES викликається перелік електричних зв'язків, який заноситься в файл з розширенням *.NLT.

Якщо креслення схеми, яку проектуємо, розташовано на кількох сторінках або вона включає в себе ієрархічні структури, то файли з розширенням *.NLT, які вміщують перелік з'єднань окремих сторінок або підсистем, об'єднуються за допомогою програми PCLINK в файл з розширенням *.XNL. Крім того, якщо в схемі компоненти описані мовою функціонально-логічного моделювання, то необхідно скласти текстовий файл з розширенням *.PML і за допомогою програми PCMODEL перетворити його у двійковий файл з розширенням *.MDL. Після цього викликається програма PRESIM для складання опису цифрового пристрою, який моделюємо, в форматі програми PCLOGS. Вихідний файл програми PRESIM за замовчуванням має те саме ім'я, що і вхідний файл, і розширення *.NET. Файл з розширенням *.NET є вхідним файлом програми моделювання PCLOGS.

Після переходу в початкове меню в режимі моделювання на екрані з'являється підказка →, яка свідчить про те, що програма очікує введення команд. Подальша робота з програмою можлива в інтерактивному або командному режимі.

Командний файл створюється за допомогою будь-якого текстового редактора або утворюється перейменуванням і редагуванням файла PCLOGS.OUT.

В командному файлі використовуються дві безрозмірні цілочисельні одиниці вимірювання часових інтервалів: крок (*time step*) і цикл (*cycle*).

Крок – це мінімальний інтервал часу, який використовується для вимірювання затримок сигналу.

Часові кроки – ціле число часових кроків, визначене в межах від 1 до 30000.

Цикл складається з кількох кроків. У синхронних пристроях це часто період тактової частоти. Тривалість сигналу може задаватися як в циклах, так і в кроках. Частіше визначають тривалість сигналів в циклах.

Наведемо перелік команд, які входять в командний файл для програми PCLOGS <ім'я файла>.cmd:

LOAD <ім'я файла>.NET
CYCLE 96
GEN [0 0] X1 (SO/1 S1/1)
GEN [0 0] X2 4(SO/2 S1/2) SO/1 S1/2
GEN [0 0] X3 2(SO/4 S1/4) SO/2
GEN [0 0] X4 (SO/8 S1/8)
GEN [0 0] XN (SO/16 S1/16)
DISPLAY 4
MONITOR X1 X2 X3 ... XN Yl Y2 .. .YM
PROBE X1 X2 X3 ... XN Y1 Y2 ... YM
SPOOL ALL
SIM 32

Спочатку за допомогою команди **LOAD** завантажується файл <ім'я файла>.NET, потім командою **GEN** задаються часові діаграми вхідних сигналів X1, X2, X3,...,XN. При моделюванні необхідно перебрати 2^N комбінацій вхідних сигналів. Наприклад, команда GEN [0 0] XI (SO/1 S1/1) означає, що вузол XI протягом одного циклу має логічний стан "0", а протягом наступного циклу – "1", після чого вони циклічно повторюються, так як взяті в круглі дужки. Затримка переднього і заднього фронтів сигналів, яка вказана в квадратних дужках, дорівнює нулю.

Після цього за командою **DISPLAY** встановлюється режим графічного виведення на екран часових діаграм вузлів, перелічених у команді PROBE. Команда DISPLAY має параметри масштабування часових діаграм. В лабораторних роботах приймається формат команди такий:

DISPLAY n/p,

де n – період показу результатів на екрані (в часових кроках);

p – параметр масштабування, який дозволяє стиснути часову діаграму за часом (за замовчуванням p=0).

Команди **SPOOL ALL** і **MONITOR** забезпечують передачу рузультатів моделювання в файл <ім'я файла>.SPL, який можна буде пізніше подивитись за допомогою програми POSTSIM.

Команда **SIM** використовується для задання тривалості моделювання в циклах. Власне моделювання починається після подачі команди SIM. В результаті на екрані будуть одержані результати моделювання.

В лабораторних роботах використовується пакетний режим завантаження кількох програм системи проектування PCAD. Для цього за допомогою текстового редактора створюється файл з розширенням *.BAT, в який заносяться командні рядки виклику послідовності програм. Зміст пакетного файла залежить від структури принципової схеми, яку досліджуємо.

Наприклад: <ім'я файла>.bat

pcmodel % 1.pml	pcnodes % 1.sch	pcnodes % 1 .sch
penodes % 1.sch	presim % 1 .nlt	pclink
presim % 1.nlt	pclogs % 1.cmd	presim % 1.nlt
pclogs % 1.cmd		pclogs % 1.cmd

Моделювання завершується командою EXIT.

7.4 Вивід часових діаграм на принтер

Після перевірки на достовірність часових діаграм, тобто перевірки на правильність функціонування схеми, необхідно вийти з режиму моделювання. Викликати пакетний файл PLOT.CMD та внести зміни. Він має вигляд:

Load e:\pcad\workp\ім 'я розділу\ім'я файла.SPL

monitor (ввести через пропуск параметри виведення на друк в необхідній послідовності).

plot *n/p* (зміст параметрів *n* і *p* такий же, як у команді DISPLAY)

review from 0

Командний файл завантажується програмою POSTSIM за допомогою командного рядка

→ POSTSIM PLOT.CMD

Командою review from 0 часова діаграма виводиться на принтер.

Робота з програмою POSTSIM завершується командою EXIT.

7.5 Вивід креслення схеми на принтер

Після створення графічного зображення схеми за допомогою програми PCCAPS усю схему цілком (або її частину) можна вивести на принтер. Для цього, знаходячись у програмі PCCAPS, командою **SYS/PL0T** зображення схеми або її частину замикають в прямокутну рамку, вказуючи координати її протилежних кутів за запитами: *PLOT: Select Page Corner 1... PLOT: Select Page Corner 2...* Після цього за запитом:

Plot file name: вводять ім'я файла, якому за замовчуванням надається розширення *.PLT.

Командою **SYS/PLOT** система зберігає всю зону креслення, яку видно на екрані, і координати вікна друку, визначені користувачем. При

виведенні даних на принтер система виконує перебудову креслення для його розміщення в указаному вікні, на що потрібен додатковий час. Тому для скорочення витрат часу при виведенні фрагмента креслення рекомендується перед виконанням команди **SYS/PLOT** виконати команду **VWIN**, яка перебудовує виділений фрагмент креслення на весь екран. Крім того, слід мати на увазі, що виводиться тільки та інформація, яка видима на екрані.

Креслення схеми виводиться на принтер за допомогою програми PCPRINT, яку можна викликати безпосередньо командою → *pcprint.*

Після цього на екран виводиться початкове меню програми:

Configure PC-PRINT (установлення конфігурації).

<<Plot a file>> (виведення креслення).

Exit PC-PRINT (вихід з програми).

Вибирається в даному меню програми виведення креслення на принтер *plot a file*. В меню побудови креслення внести такі зміни:

Plot filename – ввести ім'я файла (за замовчуванням встановлюється розширення *.PLT).

Printer: Epson MX-80: вибирається тип принтера FX-80 клавішею "Пропуск".

Plot orientation – запит на нормальне або повернуте зображення схеми. Змінюється натиском клавіші "**Пропуск**".

Scale factor: 1,5 – максимально можливий масштаб зображення схеми на креслені. Отриманий розмір необхідно зменшити на 0,1 в зв'язку з нестандартними розмірами листа паперу.

Після встановлення усіх параметрів подається команда початку друкування (натиснути клавішу **<Enter>**).

7.6 Побудова шини

Шина (джгутове з'єднання) зображається більш товстою лінією, ніж провідник. Для побудови шини спочатку за командою **ENTR/WIRE** зображаються провідники від передбаченого розміщення шини до контактів компонентів (встановлюється шар **WIRES**, товщина лінії **W:0**). Після цього командою **NAME/NET** виводять імена провідників (ланцюгів), які підводять до шини; однойменні провідники іменують однаковими іменами. Зображення шини наносять на креслення командою **DRAW/LINE** (шар **BUS**, товщина лінії **W:8**).

При цьому слід мати на увазі, що в системі PCAD шина є лише графічним зображенням на кресленні, а електричні зв'язки утворюються в результаті дослідження кожної пари зв'язаних провідників з однаковими іменами. При введенні зображення шини можна використовувати команду **ENTR/BUSB**.

7.7 Робота в середовищі PC-CAPS

7.7.1 Створення символічного опису компонента для принципової схеми

Робота виконується засобами підсистеми PC-CAPS.

Послідовність дій:

1. Встановлення робочих параметрів:

• У режимі **SYMB** командою **ZIN** встановити необхідний масштаб, для цього встановити курсор у центр екрана, натиснути клавішу 1 двічі;

• командою **VLYR** встановити параметри шарів:

GATE	ABL A
PINNUM	ABL
PINCON	ABL
REFDES	ABL
DEVICE	ABL

інші параметри в стані OFF (шари виключені).

У загальному випадку програма PC-CAPS містить такі шари:

WIRES	– електричний зв'язок;
BUS	– електричний джгут;
GATE	– зображення символьного елемента;
PINFUN	– функція виведення елемента;
PINNUM	– номер виведення;
PINNAM	– ім'я виведення;
PINCON	– шар виведення;
REFDES	– шар конструкторських позначень (мнемонічних імен);
ATTR	– шар атрибутів;
SDOT	– шар з'єднань зв'язків;
DEVICE	– функція елемента;
NETNAM	– імена ланцюгів;
CMPNAM	– імена компонентів (символьних відображень);
BOURDR	– границя схеми.

2. Побудова графічного зображення компонента:

• викликається одна з команд меню **DRAW** і в рядку статусу встановлюються параметри:

GATE	– активний шар;
SOLID	– суцільна лінія;
ORTH	– перпендикулярні відрізки (тільки для команди LINE);
W:0	– товщина лінії;
10:10	– масштаб в одиницях бази даних (DBU);
S,G	– зелений.

• командами меню **DRAW** будується графічне зображення компонента, для корекції використовуються команди меню **DEL** і **EDIT**.

3. Позначення контактів:

- команда **ENTR/PIN;**
- в рядку статусу встановити параметри:

PINCON – активний шар;
INPUT (OUTPUT, I/O) – призначення контакта;
10:10 – масштаб;
S,G – зелений.

Для кожного контакта, що вводиться, попередньо встановлюється параметр його призначення:

(INPUT, OUTPUT, I/O)

- в обрану для контакта точку підводиться хрестик, місце розташування фіксується натисканням клавіші 1мишки;
- натиснути клавішу 2, ввести з клавіатури ім'я контакта, позначений контакт висвічується жирною точкою.
- аналогічно позначити всі контакти компонента;

4. Введення текстових позначень:

- команда **DRAW/TEXT**. В рядку статусу встановити параметри:

DEVICE – активний шар;
15 – розмір букв тексту;
CCF – параметри розміщення тексту;
M – червоний;
10:10 – масштаб;
S,G – зелений.

- встановити курсор у початок тексту;
- ввести текст з клавіатури.

5. Задання ключової точки:

- команда **ENTR/ORG;**
- ключова точка (за нею відбувається виклик компонента на принципову схему), зазвичай, вибирається лівий нижній контакт.

6. Введення інформації про упакування вентиля в корпусі:

- команда **SCMD/PNLC**;
- у відповідь на підказку ввести з клавіатури кількість вентилів у корпусі;
- ввести з клавіатури кількість контактів (ніжок) на вентиль;
- встановити параметри рядка статусу:

REFDES – активний шар;
25 – розмір букв тексту;
CCF – параметри розміщення тексту.

- встановити курсор для введення конструкторського позначення (мнемонічного імені), натиснути клавішу 1. В рядку статусу встановити параметри:

PINNUM – активний шар;
15 – розмір букв тексту;
LBF – для лівих контактів;
RBF – для правих контактів.

- установкою курсору призначити місце позначення номера контакту;
- повторити для всіх контактів, при цьому висвічується кожен контакт;
- у відповідь на підказку ввести з клавіатури номер контакту (ніжки) для заданого імені;
- повторити попередню операцію для всіх контактів компонента;
- у команді **SCMD/SCAT** задати код ідентифікатора типу компонента ID ;
- команда **SCMD/SPAT**;
- перевірити і, за необхідності, із клавіатури змінити призначення контактів:

(INPUT=0; OUTPUT=1; I/O=2).

7. Збереження створеного опису:

- у команді **FILE/SAVE** ввести з клавіатури ім'я файлу. Запам'ятовується файл із розширенням "*.SYM".
- командою **SYS/QUIT** вивантажується пакет.

7.7.2 Побудова конструкторсько-технологічного образу компонента

Викликається підсистема PC-CARDS.

Послідовність дій:

1. Установка робочих параметрів:

- режим **SYMB**
- командою **VLYR** встановити параметри шарів:

PIN ABL A
SLKSCR ABL
DEVICE ABL

інші параметри – у стані OFF. У програмі PC-CARDS також є такі використовувані шари:

PIN – шар виведення;
BRDOUT – границі шару трасування;
SLKSCR – ескіз складального креслення друкованої плати.

За заданим значенням ID система розпізнає логічну функцію, виконувану компонентом, і використовує цю інформацію при логічному моделюванні електронних схем.

DEVICE – тип компонента;
REFDES – шар конструкторських позначень;
COMP – сторона компонентів;

SOLDER – сторона друку;
INTn – внутрішні *n* шарів.

- встановити необхідний масштаб. Команда **ZIN**, натиснути клавішу 1 двічі.

2. Формування "**FOOTPRINT**":

- команда **ENTR/PIN**;
- встановити параметри рядка статусу:

PIN – активний шар;
TYPE – тип контакта (тип контактної площадки на друкованій платі);
EQUIV – еквівалентність функціонального призначення контактів (ніжок) компонента;
R – червоний;
50:50 – масштаб (у DBU);
S,G – зелений.

- задати значення **TYPE** і **EQUIV** для кожного контакта, що вводиться, де:

TYPE = 0 – наскрізний/перехідний отвір (у формуванні конструкторсько – технологічного опису не бере участі);
TYPE = 1 – перша ніжка;
TYPE = 2 – всі типи ніжок, крім "1", "землі", "живлення";
TYPE = 3 – "земля";
TYPE = 4 – "живлення";
TYPE = 5..24 – нестандартний тип ніжки (задається розробником, точніше можливостями/особливостями технологічного устаткування) Однак варто пам'ятати, що наявне технологічне устаткування працює зі слайдом у 16 масок ;
EQUIV = 0 – якщо контакт не має аналога за функціональним призначенням в даному вентилі;
EQUIV = 1,2,...n – взаємозамінним контактам вентиля присвоюються однакові номери, причому функціональні аналоги в першому вентилі мають EQUIV=1, в другому - EQUIV=2 і т.д.

- установкою курсора і натисканням клавіші 1 висвітити контакт;
- ввести з клавіатури ім'я заданого контакта;
- позначити на екрані всі контакти компонента, задаючи для кожного значення **TYPE** і **EQUIV**;

Порушувати послідовність введення контактів і їхніх імен не рекомендується. Виправити помилку можна за допомогою команди **DEL**. Якщо помилка виявлена не в останньому введеному контакті, видаляються всі введені контакти до помилкового, робота повторюється.

3. Побудова графічного зображення.

• Вибрати одну з команд меню **DRAW** і встановити параметри рядка статусу:

SLKSCR – активний шар;
SOLID – суцільна лінія;
W:0 – товщина лінії.

Побудувати необхідне зображення компонента.

4. Введення текстових позначень.

• Вибрати команду **DRAW/TEXT** і встановити параметри рядка статусу:

DEVICE – активний шар;
125 – розмір букв тексту;
F – у другій позиції;
CC – симетричне розміщення тексту.

5. Позначення ключової точки.

• Командою **ENTR/ORG** указати точку, за якою даний компонент буде викликатися в PC-PLACE і PC-CARDS, зазвичай, як ключова точка вказується перший контакт.

6. Введення інформації про упакування вентилів у корпус:

• встановити необхідний масштаб командою **ZIN**;

• у команді **SCMD/SCAT** із клавіатури ввести тип компонента відповідно до таблиці ідентифікаторів ;

• командою **SCMD/SPAT** перевіряємо значення параметрів TYPE і EQUIV (за необхідності змінити параметри TYPE і EQUIV з клавіатури);

• команда **SCMD/SPKG**;

• ввести з клавіатури кількість вентилів у корпусі;

• ввести з клавіатури кількість контактів у вентилі;

• задати імена всіх запитуваних контактів;

• встановленням курсора і натисканням клавіші вказати місце розташування кожного запитуваного контакту на зображенні на екрані.

7. Збереження створеного конструкторсько-технологічного опису компонента:

При виконанні команди **FILE/SAVE** ввести з клавіатури ім'я файлу. Утвориться файл із розширенням "*.PRT".

При занесенні роз'єма у бібліотеки рекомендується описувати їх як компонент, що складається з одно вивідних вентилів (кожен вентиль – один контакт роз'єма). Вентилі в корпусі іменуються латинськими буквами в порядку алфавіту, при великій кількості вентилів – комбінацією букв за абеткою. Наприклад:

X4/b – другий контакт роз'єма X4,
X4/aa – 27 контакт роз'єма X4.

Внесення в бібліотеку символічного зображення компонента повинно задовольняти вимогам відповідного СТП [14], а конструкторсько-технологічне відображення – РТМ [15].

7.7.3 Побудова принципової електричної схеми

Робота виконується в підсистемі PC-CAPS.

Послідовність дій:

1. Установка робочих параметрів:

- У режимі **DETL**, командою **VLYR** встановити параметри шарів:

WIRES	ABL A
GATE	ABL
PINCON	ABL
SDOT	ABL
NETNAM	ABL
CMPNAM	ABL
ATTR	ABL
REFDES	ABL

Інші параметри в стані OFF.

2. Виклик файлу формату креслення:

- Командою **FILE/LOAD** ввести файл (завантажити "форматку"). На екрані відображається рамка формату креслення, вся схема повинна розміщатися в межах цього формату. Можливі ситуації, коли принципова схема зображується на декількох "форматах" [4].

3. Розміщення компонентів:

- Командою **ENTR/COMP** з робочого довідника викликаються компоненти і розміщаються на полі "форматки" відповідно до принципової схеми. У процесі роботи можливе редагування: переміщення, видалення, копіювання, повертання компонента за допомогою команд меню: **MOVE, DEL, COPY, ROT**.

4. Побудова зображень електричних зв'язків.

- Викликати команду **ENTR/WIRE**;
- Встановити параметри рядка статусу:

WIRES – активний шар;
ORTH – перпендикулярні лінії;
W:0 – товщина лінії;
L – зелений.

- Натисканням клавіші 1 графічної "мишки" задаються точки перегибу електричного зв'язку, натискання клавіші 2 завершує введення зв'язку. Не з'єднанні з провідниками контакти компонентів відображаються на екрані хрестиками, що зникають при правильному з'єднанні. У випадку перетину провідників система виводить запит, при позитивній відповіді утворюється електричне з'єднання (точка в шарі **SDOT**), яке фіксується в базі даних. Для корегування і можливості читання

принципової схеми використовують команди меню **EDIT**. Ці команди дозволяють стирати сегменти (**DELS**), переміщати вершини (**MOVA**) і т.д.

Система PCAD має засоби представлення джгутових з'єднань. При цьому використовуються такі звертання:

- Викликати команду ENTR/WIRE;
- Встановити на рядку статусу параметри:

WIRES – активний шар;
45D – з'єднання провідників під кутом 45 градусів;
W:0 – товщина лінії;
L – зелений.

- Зобразити частину провідника від місця передбачуваного розташування джгута на екрані до контакта компонента, повторити для всіх контактів компонента вхідних у джгут.
- Команда **NAME/NET** (активним обраний шар **NETNAM**);
- Присвоїти імена електричним зв'язкам.
- Викликати команду **DRAW/LINE** і встановити на рядку статусу:

BUS – активний шар;
ORTH – перпендикулярні лінії;
W:8 – товщина лінії.

- Побудувати зображення джгута. Варто мати на увазі, що при цьому сама шина є лише графічним поданням і не об'єднує зв'язки. Електричні зв'язки вводяться командою **ENTR/WIRE**. Таким чином, злиття зв'язків відбувається за їх іменами, незважаючи на те, що лінії зв'язку один до одного недоведені. З цього випливає, що різні зв'язки в різних джгутах не повинні мати однакових імен. Корисно зв'язок іменувати самому конструктору, щоб можна було активно впливати на трасування (розставляти пріоритети трасування зв'язків, вказувати різні товщини провідників для різнойменних ланцюгів і т. д.). В іншому випадку PCAD сам іменує зв'язки за шаблоном:

UN< порядковий чотиризначний номер зв'язку >

5. Введення конструкторських позначень компонентів. Система PCAD надає користувачу можливість робити компонування вентилів і корпусів на трьох рівнях:

- Командою **SCMD/PNUM** розробником вводиться повне мнемонічне ім'я обраного компонента, задається ім'я і номер вентиля. У такий спосіб даний компонент жорстко "прив'язаний" до заданого корпуса і номера вентиля. Номер вентиля задається через символ "/" латинськими буквами, наприклад: **DD5/a, X4/ad**.
- Компонування по корпусах задає розробник, вентилі компонуються автоматично. Командою **SCMD/PNUM** задається тільки ім'я без вказання вентиля.

• Компонування корпусів і вентилів здійснюється цілком автоматично, тобто користувач не присвоює своїх позначень у схемі, і в подальшій обробці підсистема PC-PACK сама призначає наскрізну нумерацію корпусів U1,U2... і т. д., використовуючи інформацію з файлів "*.SYM" і "*.LIB". Підсистема PC-PACK повертає командний файл "*.CMD", за допомогою якого коректується принципова схема. Запуск командного файлу здійснюється з підсистеми PC-CAPS, на клавіатурі набирається:

/exe <Enter>

XXX.CMD <Enter>

При цьому відбувається автоматичне іменування всіх компонентів на принциповій схемі з урахуванням рекомендованого компонування. Далі вручну завантажується принципова схема і командою

/DETL/SCMD/PNUM

імена компонентів приводяться у відповідність з рекомендаціями додатка. Повний формат команди **SCMD/PNUM** такий:

DD1/A,

де DD1 – ім'я компонента;

A – номер вентиля.

Введене ім'я відображається в шарі **REFDES** у тому місці, що було зарезервовано для введення конструкторського позначення при створенні символьного відображення "*.SYM" даного компонента. Варто також пам'ятати, що ім'я компонента введене командою **SCMD/PNUM** (а не командою **NAME/COMP**) передається наступним пакетам системи P-CAD.

6. Збереження створеної принципової схеми.

При виконанні команди **FILE/SAVE** ввести ім'я файлу (за замовчуванням – розширення "SCH").

На диску створиться файл із розширенням "*.SCH". Для одержання твердої копії принципової схеми на друкувальному пристрої чи плотері:

- команда SYS/PLOT;
- указується контур малюнка;
- вводиться з клавіатури ім'я плот-файлу (розширення "PLT" проставляється автоматично).

7.7.4 Використання ієрархії

Метод ієрархічного проектування корисний при проектуванні складних схем, що складаються з багатьох елементів, визначаючи модулі для частин схеми. Модулі можуть бути використані для створення структурних схем.

Крок 1. Створення схеми.

Для виконання цього кроку необхідно:

- встановити середовище проектування, тобто режим **DETL**;

- розмістити компоненти;
- ввести з'єднання елементів схеми;
- назвати компоненти схеми;
- назвати зв'язки;
- запам'ятовувати робочий файл не потрібно!

Для цього самого файла виконати крок 2.

Крок 2. Створення символу для схеми.

Схемі можна поставити у відповідність графічний символ, який в подальшому буде використовувати як компонента для створення схем більш високого рівня.

Для створення символу схеми в тому самому робочому файлі необхідно:

- створити графіку умовного графічного позначення схеми в режимі **SYMB**;
- ввести вхідні і вихідні виводи;
- присвоїти символу ім'я;
- назначити прив'язку символу;
- встановити тип **ID** компонента, рівний 256 (команда **SCMD/SCAT**);
- записати файл (команда **FILE/SAVE**);
- очистити робочу пам'ять і екран (команда **FILE/ZAP**).

Крок 3. Використати створений символ

Для виконання цього кроку необхідно виконати таке:

- створити схему (див. крок 1), використовуючи як один з компонентів цієї схеми створений символ;
- зберегти файл (не очищати робочу пам'ять і екран, доки не виконаний наступний крок).

Крок 4. Контроль ієрархії.

Для виконання цього кроку необхідно виконати таке:

- встановити ієрархічне розширення "донизу" (команда **LEVEL/PUSH**);
- на підказку системи: *Select a component* ..., вибрати компонент, який повинен бути розширений;
- система видасть повідомлення: *WAIT! Pushing into <NAME>.SYM* і через декілька секунд на екрані з'явиться зображення створеного символу і відповідна йому схема;
- повернутись назад (до схеми, створеної в результаті виконання кроку 3), можна використати команду **LEVEL/POP**;
- система видасть повідомлення: *WAIT! Popping back to the parent level*, і через декілька секунд на екрані з'явиться зображення початкової схеми;
- очистити робочу пам'ять і екран.

Контрольні запитання

1. Характеристика моделювального середовища PCAD.
2. Моделювання у середовищі PCAD: створення базових елементів, побудова образу принципової схеми.
3. Моделювання у середовищі PCAD: використання ієрархії, дослідження запам'ятовувальних пристроїв.
4. Призначення і функціонування програм PCNODES, PCLINK, PRESIM і PCLOGS.

8 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ САПР

При автоматизованому проектуванні для задоволення потреби прикладних програм і підсистем САПР, а також запитів користувачів у діалоговому режимі виникає необхідність у машинному представленні даних.

Інформаційний фонд САПР – це сукупність усіх необхідних для функціонування САПР даних.

Інформаційне забезпечення САПР – це сукупність інформаційного фонду і засобів його ведення, тобто засобів створення, реорганізації даних і забезпечення доступу до них з використанням ЕОМ.

До складу інформаційного фонду входять:

- нормативно-довідкова інформація (інформація про елементи, типові маршрути проектування, верстати, інструменти і т.д.);
- тимчасово записані дані, що є результатом функціонування однієї підсистеми САПР, які потім вводяться в іншу підсистему;
- програмні модулі окремих підсистем, підпрограми для розробки керувальних програм для верстатів із ЧПУ;
- креслення інструментів і пристосувань, операційні ескізи;
- шаблони для введення інформації й оформлення документів, наприклад, технологічних карт і т.п.

8.1 Ведення інформаційного фонду на ЕОМ

Відомі три підходи до організації інформаційного фонду:

1. Розміщення даних безпосередньо в тілі програми (рисунок 8.1).

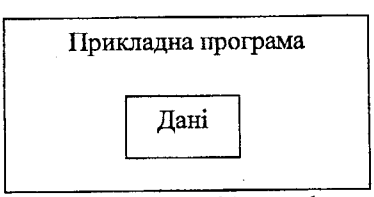


Рисунок 8.1 – Організація інформаційного фонду з розміщенням даних у тілі програми

2. Запис даних у файли (рисунок 8.2).

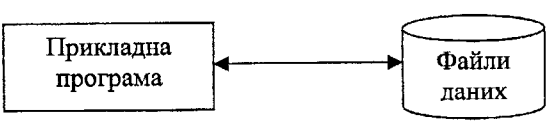


Рисунок 8.2 – Організація інформаційного фонду з записом даних у файли

3. Використання баз даних (рисунок 8.3).

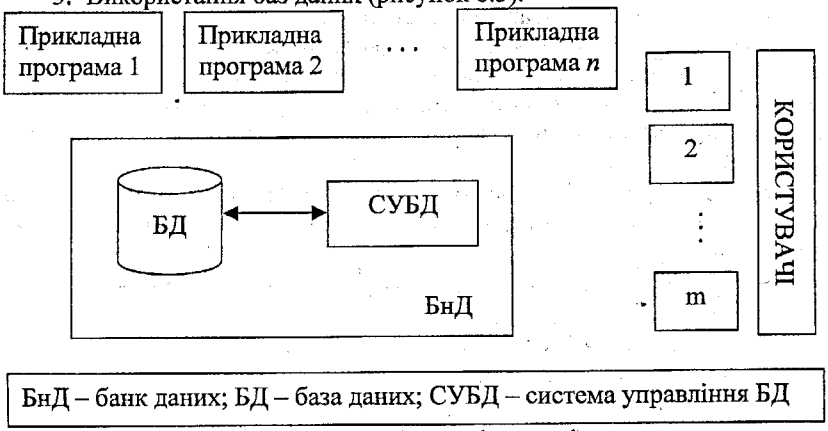


Рисунок 8.3 – Організація інформаційного фонду з використанням бази даних

У принципі, всі три підходи мають право на існування при обґрунтованому і кваліфікованому їхньому використанні в кожнім конкретному випадку.

Перший підхід у випадку, коли дані необхідно буде модифікувати, має істотний недолік: **неминучість модифікації програми** для відновлення чи реорганізації даних.

Другий підхід. При файловій організації інформація записується на вінчестер окремо від прикладної програми. Це забезпечує відносну незалежність прикладної програми від даних, тобто виключає зміну програми у випадку відновлення даних. Якщо дані використовуються тільки конкретною прикладною програмою, то такий підхід цілком прийнятний. Якщо ні, то очевидний такий недолік такого підходу: часто ті самі дані використовуються різними прикладними програмами, у яких вони мають різну структуру і подані по-різному. Це призводить до їхнього необґрунтованого дублювання (надмірності) на диску.

Є ще один недолік, що відноситься в цілому до другого підходу. До даних, розташованих в десятках файлів і організованих так, щоб задовольняти тільки запити конкретних прикладних програм, не можна звертатися користувачу, наприклад, у діалоговому режимі.

При організації інформаційного фонду з використанням запису даних у файли використовуються такі форми і методи організації і пошуку даних:

- Односторонні таблиці (матриці) рішень.
- Двосторонні таблиці (матриці) рішень.
- Алгоритмічні таблиці рішень.

- Таблиці (матриці) відповідностей.
- Логічні таблиці (матриці) відповідностей.

Третій підхід. Організація інформаційного фонду на ЕОМ з використанням баз даних (БД) застосовується в багатьох сучасних САПР ТП.

База даних – сукупність структурованих даних, використовуваних багатьма прикладними програмами і які зберігаються з мінімальною надмірністю.

Система управління базою даних (СУБД) – програмний комплекс, що забезпечує створення структури, уведення, модифікацію, видалення і пошук даних.

Іноді використовується поняття ***банку даних*** (БнД), під яким розуміється сукупність БД і СУБД.

Самою розповсюдженою в даний час є СУБД Microsoft Access, що є одним із продуктів пакета Microsoft Office.

8.2 Основні вимоги, що висуваються до баз даних

До баз даних висувається ряд вимог, серед яких можна виділити наступні основні вимоги:

1. **Мінімальна розмірність.** Кожен елемент даних вводиться в БД один раз і зберігається в одному екземплярі. При введенні даних СУБД виконує перевірку на дублювання. Цим досягається економія зовнішньої пам'яті і надійність інформації.
2. **Незалежність.** Модифікація даних і зміни, внесені в їхню структуру в зв'язку з появою нових користувачів і нових запитів, не повинні відбиватися на програмах користувачів.
3. **Цілісність даних:**

- логічна (СУБД повинна захищати БД від некоректних дій користувачів шляхом відновлення стану БД на момент, що передує помилковій операції);
- фізична (захист носіїв інформації (дисків) від збоїв шляхом дублювання, наприклад, двома паралельно працюючими дзеркальними дисками).

4. **Таємність.** Користувачі повинні працювати тільки з тими даними (фрагментами даних), до яких їм дозволений доступ.

8.2.1 Основні поняття й основи проектування баз даних

Почнемо з визначення поняття «дані».

Дані – це інформація, подана у певній формі, придатній для збереження й обробки на ЕОМ.

Можна дати й інше визначення:

Дані – це подані в цифровому вигляді відомості про деякі об'єкти навколишнього світу (про об'єкти цікавлячої предметної галузі, що нас цікавить).

При створенні будь-якої БД розробляється **модель даних**. При цьому інформація, що цікавить користувачів БД є в двох поданнях:

1. Логічне подання даних.
2. Фізичне подання даних на носії інформації (диску).

Логічне подання відбиває структуру даних. Модель не містить конкретних значень. Вона тільки описує їхню структуру. Надалі структура залишається незмінною, а дані можуть змінюватися при введенні і редагуванні інформації в БД.

Для визначення моделі використовуються такі поняття:

- об'єкт;
- атрибут;
- екземпляр;
- ключ.

Надалі вкажемо на відповідні цим поняттям поняття, використовувані при опису фізичного подання даних і поняття, прийняті в СУБД Microsoft Access.

Об'єкт – це те, про що накопичується інформація в БД, наприклад «свердел», «зенкер», «різець» і т.д.

Атрибути – це характеристики об'єкта, що цікавить користувача. Наприклад, для об'єкта «свердел» - це «позначення», «діаметр», «довжина загальна» і т.д.

Екземпляр об'єкта – сукупність значень атрибутів, що описують конкретну його реалізацію. У нашому випадку це рядок таблиці.

Ключ – це атрибут, значення якого однозначно визначає екземпляр. Так у БД по для свердлів (див. нижче) ключем може служити атрибут «позначення», тому що значення цього атрибута не дублюється в жодному рядку (екземплярі). Інші атрибути не можуть бути ключем, тому що можуть приймати однакові значення для різних екземплярів. Наприклад, цілком можливі два свердли з однаковою довжиною, хоча і різного виконання.

При описуванні фізичного подання даних, а також у термінології СУБД Microsoft Access поняттю «атрибут» відповідає поняття «поле» (стовпець таблиці). Поняттю «екземпляр» відповідає поняття «запис» (рядок таблиці). Об'єкту відповідає фрагмент файлу даних чи файл даних цілком.

База даних, що складається з набору пов'язаних між собою двовимірних (плоских) таблиць, називається **реляційною базою даних**. Дані в цих таблицях організовані таким чином, щоб забезпечити об'єднання різнорідної інформації, виключити її дублювання, а також надати оперативний доступ до наявних відомостей і ефективний супровід бази даних у цілому.

Реляційні СУБД використовують реляційну модель даних, запропоновану в 1970 році Э.Ф.Коддом. Якщо говорити спрощено, то Кодд

показав, що набір двовимірних таблиць при дотриманні певних обмежень може бути використаний для збереження даних про об'єкти реального світу і моделювання зв'язків між ними. У термінології Кодда такі таблиці називаються **відносинами** (англ. **relation**), от чому подібна база даних називається реляційною.

У реляційній базі даних, зокрема, реалізованій в СУБД Microsoft Access, для однозначного розпізнавання екземпляра об'єкта, подібно наведеному вище поняттю «ключ», вводиться унікальний ідентифікатор – **первинний ключ**.

Первинний ключ – це унікальна характеристика для кожного запису в межах таблиці. Первинний ключ таблиці крім однозначної ідентифікації записів дозволяє реалізувати і зв'язки між таблицями. Завдяки зв'язкам інформація з однієї таблиці стає доступною для іншої. Зв'язки встановлюються за рахунок того, що в різних таблицях наявні поля з однаковими значеннями.

Покажемо на прикладі переваги реляційних моделей даних перед моделями даних, побудованими на основі суцільних таблиць.

Приклад. Розробка фрагмента моделі даних «Співробітники – проекти». Нехай мова йде про технологічну службу підприємства, співробітники якої, займаючись технологічною підготовкою виробництва, звичайно, беруть участь у розробці проектів деяких виробів. Вихідні дані по співробітниках подані в так званій суцільній таблиці – див. табл. 8.1.

Таблиця 8.1

Вихідна суцільна таблиця «Співробітники – проекти»

Номер співробітника	Номер проекту	Номер завдання	Прізвище	Посада	Оклад	Відділ	Телефон
1010	AB-115	1.1	Петров	Інженер-технолог	5500	115	6-15
1010	KN-20	1.3	Петров	Інженер-технолог	5500	115	6-15
1015	ZT-14	5.2	Васильєв	Інженер-технолог	5500	115	6-15
1036	ZT-14	5.4	Куликов	Технік	3000	110	5-46
2122	AK-177	1.2	Зорин	Начальник відділу	6500	105	6-88
2122	BC-18	3.6	Зорин	Начальник відділу	6500	105	6-88

У даній таблиці є такі недоліки в поданні даних:

1. Дублюється інформація про співробітників, тому що співробітник може брати участь у декількох проектах.

2. У таблицю не можна вписати тих співробітників, що не зайняті в проектах саме зараз, а також співробітників, що взагалі не працюють над проектами, інакше таблиця не буде суцільною.

3. Якщо співробітник звільняється, запис про нього видаляється з таблиці, а разом з нею – і проект, хоча робота над ним повинна продовжуватися.

Даний приклад є класичним, оскільки відзначені вище проблеми характерні для всіх суцільних файлів.

Зняти зазначені обмеження і дозволяють реляційні бази даних. У них при проектуванні таблиць і визначенні зв'язків використовують формалізовану процедуру, що називається *нормалізацією* і спирається на апарат теорії множин і реляційної алгебри. *Нормалізація* – це покроковий процес заміни однієї таблиці іншими, що мають простішу структуру. На кожному кроці (етапі) нормалізації таблиці приводяться до деякого певного вигляду, що називається *нормальною формою*.

Щоб усунути зазначені вище недоліки, розіб'ємо вихідну таблицю на дві: «Проекти» і «Співробітники» - табл. 8.2 і 8.3.

Таблиця 8.2

Проекти		
Номер співробітника	**Номер проекту**	**Номер завдання**
1010	AB-115	1.1
1010	KN-20	1.3
1015	ZT-14	5.2
1036	ZT-14	5.4
2122	AK-177	1.2
2122	BC-18	3.6

Таблиця 8.3

Співробітники					
Номер співробітника	**Прізвище**	**Посада**	**Оклад**	**Відділ**	**Телефон**
1010	Петров	Інженер-технолог	5500	115	6-15
1015	Васильєв	Інженер-технолог	5500	115	6-15
1036	Куликов	Технік	3000	110	5-46
2122	Зорин	Начальник відділу	6500	105	6-88

Продовжимо аналіз і подивимося на таблицю «Співробітники». Нескладно помітити такі особливості:

1. Дублюється інформація про телефони для співробітників одного відділу.

2. Якщо змінюється телефон відділу, необхідно змінювати його у всіх співробітників відділу. Аналогічна ситуація буде спостерігатися при зміні розміру окладів.

3. Не можна включити дані про новий відділ, поки не будуть набрані його співробітники.

4. При звільненні усіх співробітників не зберігаються дані про самий відділ.

Тому, дотримуючись правил нормалізації, необхідно виконати декомпозицію таблиці «Співробітники» і розбити її на три таблиці: «Співробітники», «Посади», «Відділи» - табл.8.4, 8.5, 8.6.

Таблиця 8.4

Співробітники (остаточна таблиця)

Номер співробітника	Прізвище	Посада	Відділ
1010	Петров	Інженер-технолог	115
1015	Васильєв	Інженер-технолог	115
1036	Куликов	Технік	110
2122	Зорин	Начальник відділу	105

Таблиця 8.5

Посади

Посада	Оклад
Інженер-технолог	5500
Технік	3000
Начальник відділу	6500

Таблиця 8.6

Відділи

Відділ	Телефон
115	6-15
110	5-46
105	6-88

Таким чином, остаточно сформовані чотири пов'язані між собою таблиці: «Проекти», «Співробітники», «Посади» і «Відділи».

Покажемо тепер логіку подання фрагмента даних у БД «Різальні інструменти», а більш докладно в її розділі «Свердли». Інформація про свердли береться з довідника – див. табл.8.7.

Після визначення типу даних (тут застосовується два типи даних: «текстовий» – позначений нижче буквами «З» і «числовий» - буквами «N») і довжини полів виходить логічна модель даних – рисунок 8.5.

Таблиця 8.7

Довідкова інформація про свердли						
Позначення	**Діаметр, мм**	**Довжина загальна, мм**	**Довжина різальної частини, мм**	**Код хвостовика**	**Матеріал**	**ДЕРЖСТАНДАРТ**
...	...	...	...	...	...	...
65	19,00	238	135	Морзе 2	Р6М5	10903
66	19,25	238	140	Морзе 2	Р6М5	10903
...	...	...	...	...	...	...

Загальна структура даних показана на рисунку 8.4.

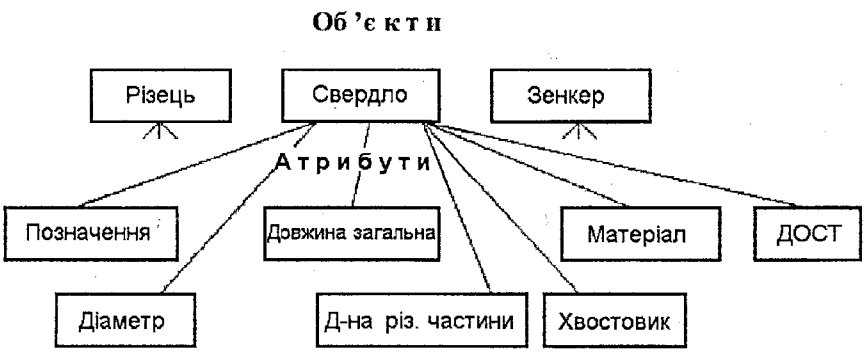


Рисунок 8.4 – Загальна структура даних БД «Різальні інструменти»

Поля запису

DB	D	L	LR	KX	KM	GOST	
CCC	NN.NN	NNN	NNN	CCCCCCC	CCCC	CCCCC	
...	...	...	...	...	...	...	...
065	19.00	238	135	Морзе 2	Р6М5	10903	←запис i
066	19.25	238	140	Морзе 2	Р6М5	10903	←запис i+1
...	...	...	...	...	...	...	...

Рисунок 8.5 – Логічна модель даних у БД «Різальні інструменти» у розділі «Свердла»

Через те, що "левина" частка роботи з проектування технологічних процесів припадає на роботу з даними і при цьому переробляється дуже

велика кількість інформації, ряд САПР ТП побудований на основі наявних СУБД. Це значно полегшує створення прикладного програмного забезпечення САПР.

Контрольні запитання

1. Інформаційне забезпечення САПР.
2. Бази даних та знань.
3. Склад і функції інформаційного забезпечення САПР.
4. Вимоги до організації даних в САПР.
5. Використання штучного інтелекту у САПР.

9 ЛІНГВІСТИЧНЕ ЗАБЕЗПЕЧЕННЯ САПР

Створення САПР є складною науково-технічного проблемного, для розв'язання якої потрібні великі матеріальні і часові витрати. Значний, а іноді і визначальний, внесок у ці витрати вносить розробка програмного забезпечення (ПЗ). Про складність ПЗ САПР говорять обсяги програм його окремих підсистем. Так, спеціальне ПЗ підсистеми функціонально-логічного проектування МОДИС-ВЕС складається з 30 тис. операторів мови ПЛ/1; ПЗ підсистеми схемотехнічного проектування САМРИС-2 – 31 тис. операторів мови ФОРТРАН (трудомісткість створення програм цієї підсистеми склала близько 20 людино-років). Розробка ПЗ підсистеми проектування компонентів у САПР ВІС, що складається з 30 програм, здійснювалася колективом з 10 висококваліфікованих фахівців протягом 7÷8 років.

До програмного забезпечення САПР висуваються вимоги, від ступеня задоволення яких значною мірою залежить ефективність використання всієї системи.

ПЗ є ланкою, що пов'язує технічне, математичне, лінгвістичне й інформаційне забезпечення в єдину проектувальну систему.

Рішення, прийняті при розробці ПЗ, тісно пов'язані з особливостями вхідних мов, бази даних, математичних методів і алгоритмів, характеристик використовуваних ЕОМ і периферійного обладнання. Далі розглядаються найзагальніші зведення про лінгвістичне забезпечення і бази даних САПР.

9.1 Класифікація мов САПР

Мови, що складають лінгвістичне забезпечення САПР, поділяються на дві групи:

1. *Мови програмування*;
2. *Вхідні мови* або *мови проектування*.

Мови програмування служать для записування програм, які складають пакети прикладних програм для автоматизації проектування.

Ними користуються, головним чином, при розробці, а не при експлуатації ПЗ.

Вхідні мови призначені для задання ЕОМ вихідної інформації для виконання проектних операцій і процедур за допомогою наявного програмного забезпечення.

Ці мови застосовуються користувачами САПР у процесі їхньої інженерної діяльності.

9.1.1 Мови програмування

У САПР як мови програмування використовуються машинно-орієнтовані мови типу мови Асемблера, алгоритмічні мови високого рівня ФОРТРАН, Delphi, Visual Basic, Visual Си, мова ЛЯПАС, орієнтована на програмування логічних і логіко-комбінаторних задач.

Алгоритмічні мови високого рівня порівняно з машинно-орієнтованими мовами: зручніші для реалізації алгоритму чисельного аналізу, легше освоюються інженерами і дозволяють підвищити продуктивність праці програмістів; дозволяють створювати програми, відносно легко переносяться на різні типи ЕОМ.

Для мови Асемблера характерні: універсальність, що виражається у великих можливостях описання кодів різних форматів, логічних операцій і процедур; менші витрати машинного часу та пам'яті при виконанні об'єктних програм, одержуваних після трансляції з машинно-орієнтованої мови.

Звичайне застосування мов високого рівня призводить до зростання витрат машинного часу і пам'яті, ступінь цього зростання залежить від якості транслятора з алгоритмічної мови і характеру алгоритму. Для більшості трансляторів з мови високого рівня проектні програми, що виходять, дають збільшення витрат машинного часу в 5÷10 разів, а з мови ФОРТРАН – у 1,5÷2,5 рази порівняно з програмами, записаними мовою Асемблера.

Прикладами використання мов програмування служать застосування мов: ФОРТРАН для створення ПЗ в САПР ВІС; у підсистемі синтезу топології МОП ВІС; у комплексах програм схемотехнічного проектування АРОПС і ПОПИТ, ЛЯПАС – у підсистемі ПРОЛОГ логічного проектування цифрової апаратури.

Але частіше намагаються використовувати переваги алгоритмічних мов високого рівня і мов Асемблера, застосовуючи одночасно при розробці САПР і ті, і інші. Так, система функціонально-логічного проектування МОДИС-ВЕС реалізована за допомогою мов ПЛ/1 і Асемблера; ті самі мови застосовані при створенні підсистеми проектування топології ВІС. При цьому мову Асемблера використовують:

а) при розробці модулів з великою кількістю логічних операцій і операцій над окремими групами розрядів машинних слів, тому що в цій ситуації можливості алгоритмічних мов високого рівня недостатні;

б) при жорстких вимогах до модуля щодо показників витрат машинного часу і пам'яті. В інших випадках визначальними стають вимоги високої продуктивності праці програмістів і інваріантості до типів ЕОМ, що обумовлює застосування мов високого рівня.

9.1.2 Вхідні мови

Вимоги до вхідних мов:

- універсальність – можливість описання вхідною мовою будь-яких об'єктів, на проектування яких орієнтована САПР;
- зручність – мова повинна мати проблемну орієнтацію, забезпечуючи користувачу максимальні зручності для описання і сприйняття використовуваних при проектуванні даних;

- максимальна лаконічність описання;
- однозначність тлумачення елементів і конструкцій мови;
- можливість розвитку і розширення мови.

Вхідні мови застосовуються користувачами САПР для описання вихідних даних у задачах проектування. У більшості випадків вихідні дані містять у собі опис об'єкта проектування й опис завдання на проектування. Відповідно у вхідній мові можна виділити підмножини, називані *мовами опису об'єкта* (МОО) і *мовами опису завдань* (МОЗ).

За допомогою МОО задається структура об'єкта при розв'язанні задач аналізу. Але і при розв'язанні задач синтезу часто потрібно задання деякої вихідної структури.

МОЗ за своїми властивостями нагадує мову керування заданнями в операційних системах, відрізняючись конкретною проблемною орієнтацією. З її допомогою задається *маршрут проектування,* під яким розуміється упорядкована послідовність задач, розв'язуваних на певному етапі проектування.

Мови опису об'єкта поділяють на:

- *процедурні* (алгоритмічні);
- *автоматні* (схемні).

Процедурні мови служать для описання процесів, що протікають у проектованих об'єктах. Зокрема, процеси можуть бути інформаційними, тоді мова виявляється засобом описання алгоритмів, реалізованих у проектованих об'єктах. Процедурні мови досить поширені в САПР ЕОМ, тому що тут вихідна інформація часто є алгоритмом функціонування об'єкта. Вони близькі за своїми основними властивостями до алгоритмічних мов високого рівня (у них можна виділити такі елементи, як оператори, арифметичні і логічні вирази, ідентифікатори змінних і т.п.).

Використовуються два різновиди процедурних мов:

- перша, називана *мовою типу АЛГОРИТМ*, призначена для опису алгоритмів функціонування без прив'язки до конкретної схеми, що реалізує алгоритм, тобто використовується для задання вихідної інформації в підсистемах синтезу;
- друга, називана *мовою типу СХЕМА*, служить для опису функцій конкретного об'єкта. Цей опис має форму алгоритму, але оператори й ідентифікатори опису виражають конкретну схемну прив'язку, що дозволяє вважати мову типу СХЕМА проміжною між мовами процедурного й автоматного типів. Мови типу СХЕМА, зазвичай, використовуються для описування вихідної інформації в підсистемах аналізу.

Автоматні мови орієнтовані на описання структури об'єкта. Запис цими мовами складається з окремих пропозицій, кожна з який служить для задання чергового елемента структури, пропозиції, вказується тип елемента, його ім'я (номер) у складі об'єкта, зв'язки з іншими елементами структури, а також описуються основні властивості у вигляді списку

значень чи параметрів посилання на бібліотеку, де зберігаються ці значення. Автоматні мови застосовуються як МОО в САПР ІЄТ, а також у ряді програм (раціонально-логічного і системного проектування ЕОМ, особливо на логічному підрівні).

Вибір процедурних чи автоматних мов для програм аналізу пов'язаний з можливостями автоматичного одержання математичних моделей аналізованих об'єктів. Для аналізу обчислювальних систем і ЕОМ при системному проектуванні, зазвичай, застосовуються математичні моделі, що складаються користувачами. У цій ситуації МОО служить для описання вже створеної моделі і тому повинна бути процедурною мовою. При аналізуванні електронних схем використовується досить розвинуте математичне забезпечення для автоматичного одержання математичних моделей. Тому в підсистемах схемотехнічного проектування МОО служить для описання структури принципових схем, тобто вона повинна бути автоматною мовою. Автоматні мови застосовуються також у підсистемах проектування компонентів і конструкторського проектування.

У практиці проектування на різних рівнях часто виникає необхідність аналізу спільного функціонування декількох блоків системи. При цьому для одержання математичних моделей прийнятної розмірності деякі з блоків повинні відображатися не докладно, а інші – приблизно. Так, для однієї частини системи може знадобитися опис структури, а для іншої частини – опис алгоритму функціонування. Використання вхідної мови, що має засоби процедурного й автоматного опису, зручно й у випадках, коли частина об'єкта вже синтезована і є його структура, а інша частина виражена ще тільки у формі алгоритму. МОО з засобами описання структур і алгоритмів зручна на етапі НДР (у системах автоматизації наукових досліджень і експериментів). Іноді для дослідження і відпрацьовування математичних моделей нових елементів в такій процедурно-автоматній мові для описання алгоритмів використовується яка-небудь із широко відомих алгоритмічних мов високого рівня. У цьому випадку вхідна мова називається *розширювальною мовою.*

Можна відзначити, що мови описання завдань – процедурні мови.

У САПР обмін інформацією між людиною й ЕОМ носить двосторонній характер. Інформація, одержувана з ЕОМ, звичайно має форму таблиць, графіків, креслень на папері чи в закодованому вигляді на перфострічці. Іноді правила подання інформації на виході ЕОМ поєднують з назвою «вихідна мова», але частіше ці правила не розглядають як самостійну мову, включаючи їх як підмножину у вхідні мови.

Вхідна мова, що поєднує засоби описання повідомлень, що йдуть у діалоговому режимі роботи як від людини до ЕОМ, так і в зворотному напрямку, називають ***діалоговою мовою,*** а вхідна мова, яка призначена для обміну інформацією з ЕОМ у пакетному режимі, – ***пасивною мовою.***

З розвитком техніки на кожному ієрархічному рівні можуть з'являтися нові типи елементів, у тому числі й елементи, що використовують нові фізичні принципи роботи. У цих умовах можуть знадобитися істотні зміни в лінгвістичному забезпеченні САПР. Однак створення нових мов і особливо трансляторів з них силами користувачів САПР – надзвичайно складна задача. Створення САПР, відкритих відносно вхідних мов, виявляється більш простим на основі введення уніфікованих *внутрішніх* (проміжних) *мов*. Схему трансляції з декількох вхідних мов на машинну мову через деяку проміжну називають *зірковою трансляцією*. Так, у САПР ІЄТ внутрішню мову можна представити як мову описання еквівалентних схем, що складаються з двополюсників п'яти типів. До таких еквівалентних схем можуть бути зведені будь-які принципові схеми, до складу яких входять різноманітні елементи.

Вхідна мова, як мова описання принципових схем, менш універсальна, але зручніша для користувача. Тому в основу підсистеми схемотехнічного проектування можна покласти інваріантну внутрішню мову і програму-компілятор. Відкритість САПР забезпечується тим, що включення в лінгвістичне забезпечення нової вхідної мови потрибуватиме розробки тільки порівняно нескладного транслятора з вхідної на внутрішню мову. При цьому те саме програмне забезпечення може використовуватися з різними вхідними мовами, для аналізу явищ різної природи (наприклад, електричних, теплових, механічних процесів). Однак, створюючи відкриту, стосовно лінгвістичного забезпечення САПР, мову не слід забувати, що єдність вхідної мови, чи, принаймні, мінімально-можлива кількість вхідних мов – позитивна властивість САПР.

9.1.3 Графічні можливості мов САПР

Графічні мови служать для задання креслень різного виду, наприклад, структурних, функціональних і принципових електричних схем. Для описання геометрії елементів креслення за допомогою графічних мов використовують такі способи:

1. Задання координат усіх граней об'єкта. Наприклад, при кодуванні плоских малюнків указуються координати всіх вузлових точок.

2. Зображення компонентів за допомогою типових графічних елементів (лінії, прямокутники, окружності, еліпси і т.п.). Цей спосіб використовується, якщо креслення задається сполученням типових фрагментів, що можуть мати досить складну форму, наприклад, графічні образи функціональних блоків ЕОМ (мікропроцесора, пам'яті, регістрів, лічильників і т.п.). При цьому спочатку однократно виконується робота з виявлення таких фрагментів, є описання і включення в бібліотеку типових компонентів.

3. Задання елементів графічного зображення у вигляді математичних співвідношень.

4. Спосіб поточкового (мозаїчного) введення, коли малюнок набирається з дрібних точок. Наприклад, розглядаються матриці розміру $n \times m$, елементи яких a_{ij} – k-значні величини, що вказують колір або градації яскравості зображення. Для цих цілей можуть використовуватися не тільки точки, але і деякі псевдографічні елементи.

Для підготовки зображення схеми на екрані дисплея використовуються спеціальні програми, називані *схемними графічними редакторами.* У таких програмах створюються графічні образи компонентів усіх бібліотек. Якщо деякий образ створений, то відповідне йому зображення може бути поміщене в будь-яке місце екрана (точно так само воно може бути виведене на графопобудовник, принтер і т.п.).

Побудова схеми здійснюється шляхом розміщення її елементів на екрані, задання їхніх параметрів (затримка проходження сигналів від входів до виходів, навантажувальна здатність і т.п.), проведення всіх з'єднань між елементами, визначення зовнішніх полюсів. Фактично тут будуть присутні ті самі дані, що і для мови структурного опису. Однак можна розглядати графічне подання схеми і на функціональному рівні і використовувати його, наприклад, для розв'язання задач логічного моделювання. Так само, як для мов структурного і функціонального опису, можна виконати трансляцію графічного подання у внутрішню (машинну) форму і використовувати останню як дані для програмних засобів САПР.

Контрольні запитання

1. Які спеціальні класи процедурних мов використовуються для опису процесів проектування?
2. Для чого використовуються внутрішні і проміжні мови?
3. Для чого служать мови супроводу і керування?
4. До яких мов відносяться штучні мови лінгвістичного забезпечення САПР?
5. Що необхідно вказати для завдання опису формальної мови?

ЛІТЕРАТУРА

1. Скляров В.А. и др. Автоматизация проектирования ЭВМ. Уч. пособие для ВУЗов, – Мн.: Вышэйша школа, 1990.–350 с.

2. Норенков Н.П., Манычев Б.В. Системы автоматизированного проектирования электронной и вычислительной техники – М.: Высшая школа, 1983.–272 с.

3. Автоматизация проектирования аналого-цифровых устройств. /Под ред. Э.И. Гитиса. – М.: Энергоатомиздат, 1987.–164 с.

4. Вермишев Ю.Х. Основы АП. – М.: Радио и связь, 1988.–280 с.

5. Автоматизация схемотехнического проектирования. Уч. пособие для ВУЗов. /Под ред. В.Н. Ильина. – М.: Радио и связь, 1987.–368 с.

6. Теоретические основы САПР: Учебник для ВУЗов/Корячко В.П., Курейчик В.М., Норенков И.П. – М.: Энергоатомиздат, 1987. – 400 с.

7. Лисицин Б.М., Кривенко В.И. Технические средства и математические методы САПР. – К.: Выща школа, 1988.–192 с.

8. Системы автоматизированного проектирования. Кн.2. Технические средства и операционные системы. – М.: Высшая школа, 1986.–156 с.

9. Казенков Г.Г., Соколов А.Г. Основы построения САПР и АСТПП. – М.: Высшая школа, 1989.–200 с.

10. Автоматизация проектирования вычислительных систем. Языки моделирования и базы данных. /Под. ред. Брейера М., – М.: Мир, 1979. – 464 с.

11. Автоматизація проектування технічних засобів ОТ. Тексти лекцій. / Уклад. О.І. Гороховський, – Вінниця: ВДТУ, 2000. – 150 с.

12. Гороховський О.І. Моделювання в середовищі Active-HDL. Лабораторний практикум. Навчальний посібник. – Вінниця: ВНТУ, 2006.– 89 с.

13. Методичні вказівки до виконання курсового проекту з дисципліни "Основи автоматизованого проектування засобів ОТ" для студентів бакалаврського напряму 6.0915 – "Комп'ютерна інженерія". Уклад. О. І. Гороховський. – Вінниця: ВНТУ, 2005. – 31 с.

Навчальне видання

Олександр Іванович Гороховський

АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ

НАВЧАЛЬНИЙ ПОСІБНИК

Оригінал – макет підготовлено автором

Редактор Т.О.Старічек

Науково-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку 9.11.2006р
Формат 29,7×42¼

Гарнітура Times New Roman
Папір офсетний

Друк різографічний
Тираж 75 прим.
Зам № 2006-183

Ум. друк. арк. 9,87

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ