DOI https://doi.org/10.15407/usim.2020.01.003
УДК 681.3.006

А.М. ПЕТРУШЕНКО, канд. фіз.-мат. наук, доцент, Київський національний університет імені Тараса Шевченка, 03022, м. Київ, просп. Академіка Глушкова, 4Д, факультет комп'ютерних наук та кібернетики,
anatoly@mytaskhelper.com

ПРИНЦИП МІКРОПРОГРАМНОГО КЕРУВАННЯ ТА АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ ОПЕРАЦІЙНИХ ПРИСТРОЇВ. І

Розглядається один із можливих підходів до вирішення на базі принципу мікропрограмного керування проблеми комплексної – від постановки задачі до отримання ескізу друкованої плати – автоматизації процесу розробки операційних пристроїв. Підхід демонструється на прикладі синтезу в діалоговій трансформаційній машині – інструментаріїю алгебро-граматичних методів подання знань у різноманітних предметних областях – керуючого автомату операційного пристрою, що реалізує операцію додавання.

Ключові слова: *принцип мікропрограмного керування, дискретний перетворювач інформації, операційний пристрій, алгебра алгоритмів, алгебро-граматичний метод подання знань, діалогова трансформаційна машина.*

Вступ

Робота виконана в рамках досліджень зі створення так званих трансформаційних методів синтезу знань у довільних предметних областях, що займають проміжне становище між класичними індуктивними і дедуктивними методами синтезу [1–9]. Загальна ідея трансформаційного синтезу [5] може бути формалізована, зокрема, в алгебро-граматичній моделі, що отримала назву граматик структурного проектування [6]. В основі цієї моделі лежать запропоновані В.М. Глушковим поняття дискретного перетворювача інформації та системи мікропрограмних (алгоритмічних) алгебр [1–7] — ефективного математичного апарату для моделювання систем будь-якої природи, поведінка яких може бути описана алгоритмічно; проте, до основних застосувань цього апарату належать теорія проектування комп'ютерів і теоретичне програмування. Діалогова трансформаційна машина (ДТМ) [10–15] може розглядатися як інструментальна підтримка проектування алгоритмів й асоційованих з ними програм і апаратури у названій моделі. Початок робіт зі створення ДТМ покладено в [10, 11], де було запропоновано архітектуру і програмно реалізовану компоненту ДТМ, призначену для автоматизації аналітичних перетворень алгоритмів і асоційованих з ними програм. В [12–15] описується реалізація ряду нових функцій ДТМ, орієнтованих, головним чином, на динамічне (діалогове) конструювання алгоритмів обробки знань в довільних предметних областях. У роботі основна увага приділяється використанню цього інструментарію для автоматизації процесу проектування набільш інтелектуальної частини комп'ютерів — операційних пристроїв (ОП), до класу яких відносяться процесори, пристроїв керування зовнішніми пристроями, канали введення/виведення тощо.

Основна задача проектування операційних пристроїв, проблеми та методи їх подолання

Основна задача проектування операційних пристроїв. В процесі системного проектування комп'ютера (який спочатку розглядається як одне ціле) на певному етапі здійснюється його декомпозиція на підсистеми, внаслідок чього виокремлюються:

- номенклатура пристроїв, зокрема ОП, з яких складається комп'ютер;
- спосіб сполучення пристроїв (інтерфейси);
- вимоги до швидкодії пристроїв.

При цьому для кожного ОП стає відомим:

- перелік входів $D = \{d_1, ..., d_H\}$ і виходів $R = \{r_1, ..., r_Q\}$, зумовлений складом інтерфейсів;
- набір операцій $F = \{f_1, ..., f_G\}$, реалізація яких покладається на ОП;
- вимога до швидкодії ОП.

З урахуванням сказаного основна задача проектування ОП формулюється так [16–19]: задано головну функцію ОП, яка визначається множинами *D, R, F* і полягає у виконанні заданої множини *F* операцій над вхідними словами із *D* з метою обчислення результатів операцій — слів із *R*, а також вимоги до швидкодії ОП. Потрібно синтезувати схему ОП, яка забезпечувала б реалізацію заданої головної функції із заданою швидкодією і була б мінімальною за кількістю використаного устаткування (як правило, остання вимога спричиняє також зменшення вартості виробу, збільшення його надійності тощо.).

Очевидно, основна задача проектування ОП потребує подальшого уточнення.

Принцип мікропрограмного керування. У будь-якому (алгоритмічно універсальному) комп'ютері всі операції реалізуються за допомогою апаратних і програмних засобів. Апаратні засоби можна умовно розділити на три основні частини: процесор, пам'ять і периферія. У процесорі також виділяються дві складові частини: пристрій керування і арифметично-логічний пристрій (АЛП). Пристрій керування розшифровує команди формування послідовності керуючих сигналів, які включають у роботу окремі вузли процесора з метою виконання дій, вказаних у команді. АЛП призначений для виконання операцій над інформацією (даними), що розглядається як складне просторово-часове перетворення машинних слів. Таке перетворення може бути описано як мікроалгоритм, що являє собою деяку послідовність елементарних машинних дій — їх називають мікрооперaціями (МО) — над словами інформації і логічних умов (ЛУ), які керують порядком слідування цих МО.

Процедура визначення структури (архітектури) ОП і порядку його функціонування, що реалізує головну функцію ОП, базується на принципі мікропрограмного керування, який, з урахуванням сказаного вище, можна описати в такий спосіб [16–19]:

- будь-яка операція $f_g \in F$ розглядається як складна дія, розділена на ряд МО;
- порядок виконання МО визначається ЛУ, які, в залежності від значень слів інформації, приймають значення "істина" або "хибність" (1 чи 0);
- процедура виконання будь-якої операцій $f_g \in F$ в ОП описується у формі мікроалгоритму, який називають мікропрограмою (МПР);
- МПР є, крім усього іншого, формою подання функції ОП. Ця функція слугує для визначення структури ОП і порядку його функціонування в часі (див. далі).

Концепція керуючого та операційного блоків. За аналогією з процесором, у структурі ОП зручно виділяти — як у структурному, так і у функціональному відношенні — дві складові частини: операційний блок (ОБ) і керуючий блок (КБ), які взаємодіють між собою згідно з принципом зворотного зв'язку. Структура ОБ служить для збереження слів інформації, виконання набору МО та обчислення значень ЛУ. МО, що реалізуються ОБ, збуджуються керуючими сигналами із деякої множини *Y*, що надходять із КБ. На станах ОБ, які виникають в ОБ після виконання МО, обчислюються значення ЛУ, які відображаються інформаційними сигналами із деякої множини *X*. Ці сигнали (значення ЛУ) поступають на вхід КБ, який під їхнім впливом генерує нові керуючі сигна-

ли, що знову надходять на вхід ОБ і так далі до завершення виконання тієї чи іншої операції. Найменування операції, яку реалізує ОП, визначається кодом операції. Як і сигнали із X, коди операцій зараховують до класу інформаційних сигналів.

Отже, КБ задає порядок виконання дій в ОБ і, тим самим, керує роботою ОП, у той час як ОБ є, по суті, виконавчою частиною ОП.

Функціонування та структуру КБ можна визначити й на базі поняття мікрокоманди – керуючого слова, що має операційно-адресну структуру та визначає порядок функціонування ОП протягом одного такту машинного часу. Послідовність мікрокоманд, що відповідає мікроалгоритму, який реалізує ту чи іншу операцію, також називають мікропрограмою цієї операції. МПР може бути записана в (постійний) запам'ятовуючий пристрій. ОП, в яких КБ реалізуються подібним чином, називають ОП з програмованою, або "м'якою", логікою. ОП, схеми яких будуються з дискретних (логічних) елементів, називають пристроями з "твердою" логікою. Існує також змішане керування операціями, при якому частина операцій реалізується на "твердій", а частина — на "м'якій" логіці. Кожен із перерахованих способів має свої переваги і недоліки. Причому останнім часом частіше використовується м'яка логіка. Проте, при вирішенні багатьох важливих практичних задач при побудові КБ зручніше використовувати "тверду" логіку керування виконанням операцій [16–19].

Зазначимо, що часто при проектуванні обчислювальної системи буває доцільним багаторівневе подання ОП. Так, в одних випадках, наприклад, при проектуванні центрального пристрою керування комп'ютером, виникає потреба розглядати АЛП як частину ОП, тоді як при проектуванні вже самого АЛП у ньому виділяються свої КБ і ОБ.

Функції операційного та керуючого блоків. На даному рівні абстракції функція ОБ задається шляхом визначення множин D, R, S, Y, X, де $D = \{d_1, ..., dH\}$ — множина вхідних слів, що вводяться в ОБ як операнди; $R = \{r_1, ..., r_Q\}$ — множина вихідних слів, що містять результати операцій; $S=\{s_1, ..., s_N\}$ — множина внутрішніх слів, що слугують для представлення інформації в процесі виконання операцій із множини $F =\{f_1, ..., f_G\}$. Припускається, що $D \subseteq S$ і $R \subseteq S$; $Y=\{y_1, ..., y_M\}$ — множина МО, що реалізують перетворення $S = \varphi_m(S)$ над словами інформації, де φ_m — обчислювана функція; $X = \{x_1, ..., x_L\}$ — множина ЛУ, де $x_i = \psi_i(S)$, а ψ_i — булева функція (БФ).

Зазначимо, що час не є аргументом функції ОБ, оскільки ОБ характеризує лише засоби, які можуть бути використані для обчислень, і ніяк не впливає на порядок протікання обчислювального процесу у часі — цю функцію бере на себе КБ.

Оскільки МО і ЛУ стосовно до КБ розглядаються як елементарні символи із множин Y і X відповідно, то його функція може задаватися операторною схемою МПР, у якій символи із Y відіграють роль операторів, а символи із X – роль ЛУ. На сьогодні задавання операторних схем алгоритмів відбувається переважно у формі граф-схем алгоритмів (ГСА) Л.А. Калужніна [20]. ГСА є графічним аналогом логічних схем алгоритмів (ЛСА) А.А. Ляпунова [21–24]. Кожна з цих форм визначає обчислювальний процес у послідовному аспекті — встановлює порядок перевірки ЛУ і порядок виконання МО. Зазначимо, при описі ГСА слова "оператор" і "мікрокоманда" будемо розглядати як синоніми, тобто під оператором будемо розуміти множину МО, що записані в одній операторній вершині і виконуються в ОП одночасно. Надалі, виходячи з функцій КБ і ОБ, визначаються структури цих блоків.

Основні характеристики операційного блоку. На базі введених понять уточнимо характеристики ОП, що входять до формулювання основної задачі проектування ОП. До основних характеристик ОП належать швидкодія і витрати устаткування [16–19]. Швидкодія ОП визначається середнім часом виконання операцій $\omega = \sum_{g=1}^{G} p_g \tau_g$, де p_g — ймовірність виконання операцій f_g; τ_g — середній час виконання операції f_g. Ймовірності $p_1, ..., p_G$ визначаються класом задач, для вирішення яких призначе-

ний ОП і які вважаються відомими. ОП функціонує в дискретному часі $t = 0, 1, 2, ...$ Проміжок між двома сусідніми моментами часу t і $t+1$ називається тактом. Протягом такту формується набір керуючих сигналів, виконуються відповідні МО й обчислюються значення ЛУ. Тривалість такту T залежить від складності МО й ЛУ та від швидкодії елементів, з яких побудовані КБ і ОБ. З урахуванням сказаного, середній час виконання однієї операції $\tau_\gamma = T*\Theta$, де T — тривалість такту; Θ — середня кількість тактів, за яке ОП реалізує операцію f_g.

Витрати устаткування в ОП: $C = C_{КБ}+C_{ОБ}$, де $C_{КБ}$ і $C_{ОБ}$ — вартість елементів у КБ і ОБ відповідно. При структурній реалізації в ОБ виокремлюють три частини: пам'ять S; комбінаційну схему (КС) Ф, що реалізує МО; КС Ψ, що обчислює значення ЛУ. Тоді $C = C_S + C_Ф + C_Ψ + C_{КБ}$, де C_S, $C_Ф$, $C_Ψ$, $C_{КБ}$ – витрати устаткування, відповідно, на пам'ять S, КС $Ф$, КС Ψ та КБ.

Вплив систем мікрооперацій і логічних умов на характеристики операційних пристроїв. Найбільш істотний вплив на характеристики ОП мають множини МО $Y = \{y_1, ..., y_M\}$ і ЛУ $X = \{x_1, ..., x_L\}$ і, насамперед, системи функцій, на основі яких вони будуються. Дійсно, згідно з визначенням, МО $y_m \in Y$ і ЛУ $x_l \in X$ синтаксично записуються як $s_\alpha := \phi_m(s_{\beta 1}, ..., s_{\beta k})$ і $x_l := \psi_l(s_{\gamma 1}, ..., s_{\gamma h})$ відповідно. Отже, у множинах Y і X елементи розрізняються як функціями $\Phi = \{\varphi_m\}$ і $\Psi = \{\psi_l\}$, на основі яких утворюються МО і ЛУ, так і наборами слів $s_\alpha, s_{\beta 1}, ..., s_{\beta k} \in S$ і $s_{\gamma 1}, ..., s_{\gamma h} \in S$, які є операндами цих функцій. Систему $Z = <Ф, Ψ>$ називають системою утворюючих алгоритму[16–19].

Виявляється [18], що кількість тактів, необхідних для виконання операції, яку реалізує ОП, залежить як від самої операції, так і складу системи Z: чим "елементарніші", простіші функції з Ф та Ψ, тим більша кількість тактів потрібна для виконання операції. Але це означає, що середній час τ_g виконання операції збільшується, а швидкодія ОП зменшується.

Крім цього, зміна системи Z шляхом розкладання операцій із F на усе більш прості операції (МО і ЛУ) приводить до зменшення витрат устаткування $C_{ОБ}$ в ОБ, але, водночас збільшуються витрати устаткування $C_{КБ}$ в КБ. Виходить, що змінюючи склад МО і ЛУ, у термінах яких описується МПР виконання операцій, можна "перекачувати" устаткування з ОБ у КБ і навпаки. При цьому можна знайти деяку систему утворюючих Z_m, яка забезпечить мінімум витрат устаткування в ОП. Однак, відповідний системі Z_m час виконання операцій ω досить великий. Щоб зменшити значення ω, тобто збільшити швидкодію ОП, необхідно додаткове устаткування, що використовується для реалізації більш складних МО і ЛУ.

Отже, система утворюючих $Z = <Ф, Ψ>$ суттєво впливає на характеристики ОП. Тому визначення набору функцій Z є першочерговою задачею мікропрограмування. Однак, задача породження систем утворюючих, що мінімізують витрати устаткування та забезпечують заданий час виконання операцій у пристрої, досить складна, що ускладнює розробку формальних методів оптимізації характеристик ОП. В.М. Глушков запропонував підхід до рішення цієї проблеми, який базується на апараті алгоритмічних (мікропрограмних) алгебр. У [3] наведено приклад перетворення мікропрограми множення двійкових чисел, який став уже класичним. Зазначимо, що апарат САА втілено у мові САА\Д (див. далі).

Концепція функціонального мікропрограмування. Щоб синтезувати структуру ОП, необхідно прийняти деякий спосіб виконання операцій в ОП й описати його у формі МПР. МПР, що представляє функцію ОП безвідносно до засобів (структури ОП), які можуть бути використані для реалізації заданої функції, називається функціональною МПР [16–19]. Функціональна МПР фіксує в собі алгоритм виконання операції, який рекомендується розробником, і, крім цього, використовується як вихідна форма подання функції ОП, на основі якої синтезується структура КБ і ОБ, достатня для реалізації цієї функції.

Зазначимо, процедура вибору розробником алгоритму реалізації тієї чи іншої операції формалізована недостатньо, й тому розробник, у загальному випадку, рекомендує алгоритм,

який, як правило, погано (не повно) враховує витрати часу й устаткування на його реалізацію, роль окремих алгоритмів у процедурі "оптимального" об'єднання відповідних їм закодованих граф-схем МПР в один граф (див. далі) тощо.

Для запису функціональних МПР необхідно мати ту чи іншу мову функціонального мікропрограмування (ФМ-мову). ФМ-мова розглядається, зазвичай, як інженерна, тобто не машинна, мова. З цієї причини для зручності в ФМ-мові використовують, зазвичай, загальноприйняту математичну символіку та графічне представлення схем алгоритмів.

Функціональні МПР містять у собі дві частини:

- опис слів і масивів, що встановлює типи та формати слів, з якими оперує МПР;
- змістовний граф МПР, який визначає алгоритм виконання операцій в ОП не в термінах елементів множин Y і X, а у вигляді описів МО і ЛУ в змістовній формі — така форма застосовується, як правило, на початкових (попередніх) етапах проектування ОП.

Розрізняють такі основні типи слів:

– вхідні — значення приписуються поза МПР, а використовуються всередині МПР;

– внутрішні (локальні) — значення приписуються та використовуються лише всередині МПР;

– допоміжні — значення приписуються та використовуються лише всередині МПР, але існують не постійно, а лише протягом обмежених інтервалів часу (в межах такту);

– вихідні — значення приписуються в МПР і використовуються поза нею.

Типи слів позначаються буквами: I — вхідні, L — внутрішні, A — допоміжні, O — вихідні. Деякі слова можуть використовуватися як елементи інформації декількох типів, тобто можуть мати, наприклад, такі типи: *IL*, *LO*, *ILO* тощо.

Деякі з МО із даної МПР можуть виконуватися паралельно в часі, тоді як інші — лише послідовно. Кажуть, що сукупність МО володіє властивістю сумісності, якщо ця властивість гарантує можливість паралельного виконання МО із цієї сукупності. МО, що не володіють зазначеною властивістю, називаються несумісними. Сумісність МО буває двох типів: функціональна і структурна. МО є функціонально сумісними, якщо вони присвоюють значення різним словам, тобто якщо $S_1 := \varphi_1(S_2)$ і $S_3 := \varphi_2(S_4)$, де S_1, S_2, S_3, $S_4 \subseteq S$, то МО φ_1 і φ_2 — функціонально сумісні, якщо $S_1 \cap S_3 = \varnothing$. Структурна сумісність зумовлена структурою ОП, яка допускає чи виключає можливість паралельного виконання декількох МО. Якщо структуру ОП не визначено, то сумісними вважаються функціонально сумісні МО, інакше — структурно сумісні МО. Очевидно, що сумісні МО можуть записуватися в одній операторній вершині й розглядатися як один оператор (як одна мікрокоманда) Y_t, $t = 1, ..., T$, де T — кількість операторних вершин МПР. Далі ми покажемо, що існують структури ОП, які забезпечують сумісність усіх функціонально сумісних МО, але більшість структур вносить обмеження на сумісність окремих груп чи всіх без винятку МО.

Отже, засоби ФМ-мови мають, перш за все, забезпечувати опис алгоритмів виконання операцій з таким ступенем деталізації, що уможливлює синтез адекватної структури ОП. Однак, це ще не все. Бажано також, щоб такі мови містили у собі засоби для перевірки коректності алгоритмів шляхом їхнього моделювання на комп'ютері та, крім цього, якісь засоби для оптимізації цих алгоритмів і асоційованих з ними ОП за тими чи іншими критеріям. На жаль, сучасні ФМ-мови недостатньо підтримують зазначені вимоги, особливо у комплексі.

Уточнення функцій операційного та керуючого блоків. Легко бачити, що функціональні МПР $M_1, ..., M_G$, які описують алгоритми виконання в ОП операцій $f_1, ..., f_G$ із F, несуть у собі всю інформацію про функції ОБ і КБ. Отож, визначимо спочатку функцію ОБ.

Нехай S_g, Y_g і X_g — множини слів, МО і ЛУ відповідно, які вводить функціональна МПР M_g, що реалізує алгоритм виконання операції $f_g \in F$, $g = 1, ..., G$. Тоді множини S, Y, X (вони задають функцію ОБ) достатні для реалізації всіх

операцій з множини $F = \{f_1, ..., f_G\}$, визначаються як об'єднання, відповідно, множин S_g, Y_g, X_g для всіх M_g, $g = 1, ..., G$.

Деякі проблеми виникають лише при знаходженні множини S. Дійсно, можна очікувати, що витрати устаткування в ОБ будуть тим менше, чим менше сумарне число розрядів міститься в словах із S. Отже, при об'єднанні слів із $S_1, ..., S_G$ необхідно вміти, якщо це можливо, ототожнювати їх між собою. Коли функціональні МПР створюються незалежно одна від одної, то для ототожнення слів потрібна спеціальна процедура. Однак у роботі ми лише припускаємо, що функціональні МПР будуються з урахуванням їх об'єднання й, у зв'язку з цим, тотожні слова, що використовуються в різних МПР, ідентифікуються однаковими іменами. Таке припущення дозволяє процедуру об'єднання множин $S_1, ..., S_G$ звести до перерахування в множині S слів з попарно різними ідентифікаторами, а якщо слова з однаковими ідентифікаторами мають різну довжину, то в множину S включати лише слово з максимальною довжиною (якщо слово меншої довжини можна подати в більшому форматі).

Множина Y визначається перерахуванням усіх попарно різних МО (операторів присвоювання) з множин $Y_1, ..., Y_G$, що "витягуються" зі змістовних МПР $M_1, ..., M_G$ відповідно. Легко бачити, що число МО у множині Y можна зменшити, якщо при розробці функціональних МПР $M_1, ..., M_G$ використовувати, по можливості, одні й ті самі МО.

Аналогічно визначається достатня для реалізації операцій із F множина ЛУ X.

Функція КБ, як ми вже знаємо, задається:

- множиною X вхідних (інформаційних) сигналів, що відбивають стан ОБ;
- множиною Y вихідних (керуючих) сигналів, що ініціюють реалізацію МО в ОБ;
- операторною граф-схемою МПР, яка задає порядок проходження керуючих сигналів із Y залежно від значень інформаційних сигналів X.

Однією з форм операторних схем є так звана закодована граф-схема МПР, яку можна отримати зі змістовної граф-схеми МПР шляхом заміни МО (ЛУ), що знаходяться в операторних (умовних) вершинах, на відповідні їм символи із множини $Y = \{y_1, ..., y_M\}$ ($X = \{x_1, ..., x_L\}$). При цьому однакові МО і ЛУ, що знаходяться у різних операторних і умовних вершинах МПР, замінюються на однакові символи із множин Y і X відповідно.

Якщо функція КБ визначається сукупністю закодованих ГСА $\Gamma_1, ..., \Gamma_G$, що відповідають змістовним ГСА $M_1, ..., M_G$, то, в принципі, на основі ГСА $\Gamma_1, ..., \Gamma_G$ можна синтезувати G КБ, що забезпечать керування ОБ. Однак таке рішення не є оптимальним з наступних причин [16–19]. ГСА різних МПР, як правило, містять однакові операторні й умовні вершини. Якщо процедуру об'єднання ГСА $\Gamma_1, ..., \Gamma_G$ в одну ГСАГ побудувати таким чином, що однакові операторні й однакові умовні вершини ГСА $\Gamma_1, ..., \Gamma_G$ будуть "зливатися" в одну операторну й одну умовну вершину ГСА Г, то можна очікувати, що число операторних й умовних вершин у ГСА Г — у порівнянні з аналогічними числом у граф-схемах $\Gamma_1, ..., \Gamma_G$ — зменшиться, що, у свою чергу, спричинить зменшення витрат устаткування в КБ. У зв'язку з цим слід очікувати, що витрати устаткування в ОП, побудованому по об'єднаній ГСА Г, у загальному випадку, будуть менші, ніж сумарні витрати устаткування в G ОП, кожен з яких реалізує відповідну закодовану ГСА $\Gamma_1, ..., \Gamma_G$. Таким чином, функцію КБ доцільно представляти у вигляді закодованої ГСА, що є об'єднанням закодованих ГСА МПР, які описують алгоритми виконання окремих операцій.

Метод об'єднання ГСА $\Gamma_1, ..., \Gamma_G$ у єдину граф-схему Г, що містить мінімальу кількість операторних і умовних вершин, наведено в [19]. В об'єднаній МПР шляхи розвитку процесу обчислень, що відповідають різним операціям $f_1, ..., f_G$, задаються h-розрядним набором двійкових змінних $p_1, ..., p_h$, де $h \geq \log_2 G$, за допомогою яких кодуються операції. При цьому змінні $p_1, ..., p_h$ відіграють роль ЛУ, які визначають переходи в об'єднаній МПР. Зрозуміло, чим більшою є кількість МПР, що беруть участь в об'єднанні, тим більше можливостей створюється для злиття фрагментів різних МПР і, як наслідок, кількість вершин в об'єднаній МПР

часто виявляється значно меншою сумарної кількості вершин в об'єднуваних МПР.

Характеристики операційного блоку. Основними характеристиками ОБ є продуктивність, швидкодія, витрати устаткування (вартість), регулярність та універсальність [16–19].

ОБ можна розглядати як самостійний об'єкт. У цьому разі такт $\tau_{ОБ}$ — це проміжок часу від моменту надходження на вхід ОБ керуючих сигналів до моменту вироблення значень інформаційних сигналів, що відповідають стану пам'яті ОБ. Тривалість T такту визначається максимальним значеням, необхідним для виконання будь-якої МО й обчислення значення будь-якої ЛУ. При цьому тактуючі (синхронізуючі) сигнали, які виробляються в комп'ютері генератором тактуючих імпульсів, слідують один за одним з періодом T.

Продуктивність ОБ — це кількість МО, що виконується в ОБ протягом такту. В одному такті можуть виконуватися лише сумісні МО, що знаходяться в одній операторній вершині МПР, і їхнє число може змінюватися від такту до такту. Тому кількість МО, що виконуються в ОБ за один такт, є дискретною випадковою величиною.

Фактор випадковості вноситься вхідними даними D, у залежності від значень яких процес виконання МПР може розвиватися різними шляхами алгоритму. З урахуванням цього продуктивність оцінюють або максимальною кількістю МО, які може виконати ОБ протягом такту, або середнім значенням. Якщо припустити, що МПР складається з K операторів, кожен із яких містить m_k сумісних МО, і що в одній реалізації МПР кожен оператор виконується у середньому q_k раз (k=1, ..., K), то середня кількість МО, що виконуються ОБ в одному такті, визначається як $V = \frac{\sum_{k=1}^{K} q_k m_k}{\sum_{k=1}^{K} q_k}$.

Швидкодія ОБ характеризується тривалістю такту $\tau_{ОБ}$ — чим менше тривалість такту, тим вище швидкодія ОБ. Швидкодія залежить як від структури КС Ф і Ψ, так і від швидкісних характеристик логічних і запам'ятовуючих елементів, з яких побудовано КС і пам'ять S ОБ.

Витрати устаткування $C_{ОБ}$ в ОБ, як і раніше, визначаються сумою: $C_{ОБ} = C_S + C_Ф + C_Ψ$.

Регулярна структура складається з однотипних частин, які однаково пов'язані між собою. Очевидно, регуляризація структури спрощує процес її виробництва, що, у кінцевому рахунку, призводить до зменшення її вартості та збільшення надійності.

Універсальність (багатофункціональність) структури ОБ проявляється в можливості реалізації однією структурою досить широкого класу функцій (алгоритмів). Якщо структура універсальна, то реалізація будь-якого алгоритму зводиться до перенастроювання структури зовнішніми засобами, наприклад шляхом програмування. Якщо система МО і ЛУ повна, то максимальний ступінь універсальності досягається, якщо будь-які МО і ЛУ можуть бути поширені на кожен з регістрів ОБ. Легко бачити, що чим більш універсальна структура, тим ширше область її застосування. Збільшення ступеню універсальності дозволяє скоротити номенклатуру виробів, збільшити об'єм їх випуску та знизити вартість виробництва.

Виявляється [18], що регулярність і універсальність — взаємозалежні властивості. Регулярні структури, як правило, більш універсальні, ніж нерегулярні, а збільшення ступеня універсальності можна досягнути шляхом регуляризації структури.

Формальні методи проектування обчислювальних систем

Дискретні перетворювачі інформації (*ДПІ*) можуть розглядатися, зокрема, як базова математична модель комп'ютерів і використовуватися як відправна точка для розв'язання задач автоматизації проектування обчислювальних систем. Цей підхід, запропонований В.М. Глушковим [1–7], узагальнює ідеї А. Тьюринга і ґрунтується на припущенні, що узагальненою моделлю будь-якого ОП служить композиція зі зворотним зв'язком двох автоматів — керуючого (КА) та операційного (ОА), причому вибирають, як правило, автомати Мілі та Мура

відповідно. Такий підхід має загальний характер і використовується не лише в моделях однопроцесорних обчислювальних систем, але й у тих випадках, коли ОП є багатопроцесорним комплексом [5, 6]. Уведемо необхідні поняття.

Розглянемо ініціальний, у загальному випадку — частковий, детермінований автомат $\mathfrak{A} = <A, Q, B, \delta, \gamma>$, де A — вхідний алфавіт, Q — алфавіт станів, у якому виділено початковий q_0 і заключний q^* стани, B — вихідний алфавіт, δ — функція переходів і γ — функція виходів. Нехай M — непорожня множина, яку називають інформаційною, і припустимо, що задані два (часткових) відображення π: $M \to A$ та ξ: $B \to F$, де $F = F_M$: $M \to M$ — сукупність часткових відображень із M в M.

Дискретним перетворювачем інформації, що діє на інформаційній множині M, називають четвірку $T = <\mathfrak{A}, \pi, \xi; M^0>$, де $M^0 \subseteq M$, відображення з F називають елементарними операторами, π — функцією відміток, M^0 — множиною початкових станів множини M, а відображення π і ξ — інтерпретацією вхідних і вихідних сигналів ДПІ T [1–5].

Припускається, що функціонування ДПІ T розпочинається у початковий момент часу, коли $\mathfrak{A}$ встановлено в стан q_0, а інформаційну множину M — у стан $x \in M^0$. Якщо $\pi(x)=a$, то $\mathfrak{A}$ під впливом a, переходить у стан $\delta(q_0, a)$, а на його виході з'являється символ $\gamma(q_0,a)=b$, якому ставиться у відповідність відображення $\xi(b) = f$. Отже, один цикл роботи ДПІ перетворює початковий стан x на стан $y = f(x)$. Далі знов обчислюється $\pi(y)$ і т. д. Процес триває доти, доки не відбудеться одна з подій: автомат $\mathfrak{A}$ потрапить у заключний стан $q^* \in Q$; відображення $f \in F$ або одна з функції π, ξ, δ, γ будуть невизначені у якийсь момент часу. Як тільки одна з цих подій відбувається, ДПІ припиняє роботу.

Описаний процес часто подають як спільне функціонування двох автоматів: КА $\mathfrak{B}$ (див. раніше) і ОА $\mathfrak{B}$, який визначається так: $\mathfrak{B} = <B, \mathcal{M}, A, \sigma, \pi>$, де B — вхідний алфавіт, A — вихідний алфавіт, $\mathcal{M}$ — множина станів, елементами якої є елементи інформаційної множини M, σ — функція переходів така, що $\sigma(x, b) = f_b(x)$, де $f_b = \xi(b)$, і π — функція виходів, що співпадає з раніше визначеним відображенням π: $M \to A$. Спільне функціонування автоматів $\mathfrak{A}$ і $\mathfrak{B}$ полягає у наступному. Задаються початкові стани КА та ОА і значення виходу ОА на його початковому стані. Спочатку спрацьовує КА $\mathfrak{A}$, значення його виходу подається на вхід ОА $\mathfrak{B}$, який змінює свій стан і подає на вхід символ із B і так само далі.

Зазначимо, на відміну від КА, який, зазвичай, є скінченним, ОА, у загальному випадку, має нескінченну множину станів, оскільки на потужність інформаційної множини, елементи якої моделюють оброблювану інформацію, ніяких обмежень не накладається.

Легко бачити, що ДПІ T задає часткове відображення r_T: $M^0 \to M$, причому $r_T(x)$ співпадає зі значенням оператора $f_b = \xi(b)$, обчисленим у заключному стані КА. Якщо автомат не потрапляє в цей стан, значення $r_T(x)$ не визначено. Значення r_T не визначено також у точках з $M \setminus M^0$, проте ці точки можуть бути проміжними і заключними станами при обчисленнях. Часткове відображення r_T називають оператором, поданим у ДПІ T.

Алгебра В.М. Глушкова. Нехай M — непорожня множина, яку, як і раніше, будемо називати інформаційною, а її елементи — станами інформаційної множини M; F_M: $M \to M$ — сукупність різноманітних часткових відображень із M в M; e — тотожне на M відображення; η — ніде не визначений на M оператор; P_M: $M \to E_3$ — множина всіх одномісних предикатів на M, що набувають значення в $E_3 = \{0, \mu, 1\}$, де μ — невизначене значення.

Уведемо операції:

- диз'юнкції φ_1: $P_M \times P_M \to P_M$, $\varphi_1(\alpha, \beta) = \alpha \vee \beta$ (тут і далі $\alpha, \beta \in P_M$);
- кон'юнкції φ_2: $P_M \times P_M \to P_M$, $\varphi_2(\alpha, \beta) = \alpha \wedge \beta = \alpha\beta$;
- заперечення φ_3: $P_M \to P_M$, $\varphi_3(\alpha) = \neg\alpha$;
- лівого множення умови на оператор (прогнозування): φ_4: $F_M \times P_M \to P_M$, $\varphi_4(f, \alpha)(x) = \alpha(f(x))$ для всякого $x \in M$; $\varphi_4(f, \alpha) = f\alpha$;
- композиції відображень ψ_1: $F_M \times F_M \to F_M$, $\psi_1(f, g) = fg$;

▪ α-диз'юнкції ψ_2: $P_M \times F_M \times F_M \to F_M$:

$$\begin{cases} f(x), \text{ якщо } \alpha(x) = 1, \\ \psi_2(\alpha, f, g)(x) = g(x), \text{ якщо } \alpha(x) = 0, \\ \eta(x), \text{ якщо } \alpha(x) = \mu. \end{cases}$$

Зазвичай, α-диз'юнкцію $\psi_2(\alpha, f, g)$ операторів f і g позначають як $_\alpha(f \vee g)$ або $[\alpha](f \vee g)$;

▪ α-ітерації ψ_3: $P_M \times F_M \to F_M$ задається так. Нехай $\alpha \in P_M$, $f \in F_M$, $x \in M$, тоді $k(\alpha, f, x) = \{n \in N: \alpha(f^i(x)) = 0, i = 0, 1, ..., n\}$, де $f^0(x) = x$. Через m позначимо $\max k(\alpha, f, x)$, якщо цей максимум існує, і $\alpha(f^{m+1}(x)) = 1$. Тоді

$$\begin{cases} e(x), \text{ якщо } \alpha(x) = 1, \\ \psi_3(\alpha, f)(x) = f^{m+1}(x), \text{ якщо } \alpha(x) = 0, \\ \qquad\qquad m = \max k(\alpha, f, x), \\ \eta(x) - \text{ в інших випадках.} \end{cases}$$

▪ α-ітерація оператора f позначається, зазвичай, як $_\alpha\{f\}$ або $[\alpha]\{f\}$.

Нехай $\Omega = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \psi_1, \psi_2, \psi_3\}$ і задана множина сортів $S = \{1, 2\}$, $A = \{A_1, A_2\}$, де $A_1 = \Phi \subseteq F_M$, $A_2 = \Pi \subseteq P_M$. Системою алгоритмічних (мікропрограмних) алгебр (САА) називають скінченно породжену двохосновну алгебру $\mathfrak{N} = \langle A, \Omega \rangle$ (або $\mathfrak{N} = \langle \Phi, \Pi, \Omega \rangle$) з фіксованою скінченною системою твірних (Φ_0, Π_0), де $\Phi_0 \subseteq \Phi$, $\Pi_0 \subseteq \Pi$. При цьому Φ називають простором операторів, Π — простором умов, оператори з Φ_0 — елементарними операторами, а предикати з Π_0 — елементарними умовами алгебри $\mathfrak{N}$. Операції сигнатури Ω цієї алгебри мають наступні типи: $\varphi_1(22, 2)$, $\varphi_2(22, 2)$, $\varphi_3(2, 2)$, $\varphi_4(12, 2)$, $\psi_1(11, 1)$, $\psi_2(211, 1)$, $\psi_1(21, 1)$.

Нехай тепер X (Y) — скінченна множина змінних, кожній з яких у взаємнооднозначну відповідність поставлено функціональний (предикатний) символ, що відповідає оператору з Φ_0 (умові з Π_0). Розглянемо множину $R_{\mathfrak{N}}$ термів сигнатури Ω, що складаються з цих змінних. Терми з $R_{\mathfrak{N}}$ називаються регулярними схемами. Задана відповідність, зазвичай, являє собою інтерпретацію змінних, які входять до регулярних схем. Разом з тим визначені і значення кожного з термів $R_{\mathfrak{N}}$. При цьому терм набуває значення в Φ (в Π) лише тоді, коли він має сорт 1 (сорт 2). Якщо t — терм сорту 1 (сорту 2), що набуває значення $f \in \Phi$ ($\alpha \in \Pi$), то регулярну схему t називають регулярною схемою оператора f (умови α) і кажуть, що оператор $f \in \Phi$ (умова $\alpha \in \Pi$) подано регулярною схемою t. САА ще називають алгоритмічною алгеброю, алгеброю алгоритмів або алгеброю Глушкова.

Надалі відповідно до класу задач, на вирішення яких був спрямований апарат САА, сигнатура Ω змінювалася: деякі операції вилучались, інші – вводились у сигнатуру; іноді до імені САА, що утворюється при цьому, додається слово "модифікована".

Апарат САА є еквівалентним за своїми можливостями обчислювати функції класичним алгоритмічним системам – таким, як машини Тьюринга, алгорифми Маркова, схеми Янова тощо. Проте, у порівнянні з цими формалізмами, САА більш зручна в практичному відношенні, оскільки за допомогою операцій її сигнатури Ω можуть бути описані "програми" обчислення значень операторів із Φ (або умов із Π), зображених за допомогою інших, елементарних, операторів і умов. Якщо "елементарні" оператори та умови вибираються з деякої фіксованої системи твірних C, то конструкції програм можна описати елементами алгебри $R(\Omega, C)$, тобто термами сигнатури Ω, в яких змінними є елементи системи твірних алгебри $\mathfrak{N}$. У зв'язку з цим регулярна схема оператора може розглядатися як математична модель при проектуванні ОП та реальних алгоритмів і програм, а також при розробці відповідних інструментальних засобів (див. далі).

Перевагою алгебр Глушкова є також можливість проведення на основі тотожностей даної алгебри (справедливих як для операторів і умов будь-якої алгебри, так і для конкретних операторів і умов конкретної алгебри) різноманітних перетворень (з метою, наприклад, оптимізації за тим чи іншим критерієм) регулярних схем як алгоритмів і програм, так і операторів (мікропрограм) даної алгебри [6].

При вирішенні задач автоматизації проектування обчислювальних систем з'являється проблема відповідності формальних засо-

бів, які використовуються для моделювання роботи комп'ютерів, і засобів, що описують "програми" обробки даних на них. До перших належить, зокрема, формальне подання комп'ютерів у вигляді ДПІ, других — апарат САА. У зв'язку з цим особливе значення має наступний результат, отриманий В.М. Глушковим [4–6].

Теорема Глушкова. Будь-який оператор, поданий у скінченному ДПІ, може бути заданий і регулярною схемою САА, асоційованою із цим ДПІ.

Зазначимо, що вивчення різних форм подання алгоритмів і розв'язання проблем їх відповідності один одному має давню історію і становить не лише теоретичний, а й глибокий практичний інтерес (див. далі).

Алгоритм синтезу операційних пристроїв

З урахуванням проблематики, що виникає у результаті аналізу основної задачі проектування ОП, і зроблених при цьому застережень, сформулюємо алгоритм синтезу ОП. Цей алгоритм включає алгоритми синтезу КА та ОА, з яких складається ОП [16—19].

Алгоритм синтезу керуючого автомата умовно розбивається на наступні етапи.

Етап 1. Для кожної з операцій $f_1, ..., f_G$, реалізація яких покладається на ОП, обирається відповідний цій операції алгоритм, який реалізовуватиме її.

Етап 2. Фіксуються мова і форма опису алгоритмів. У термінах обраних засобів — нехай це будуть граф-схеми — створюють змістовні МПР $M_1, ..., M_G$.

Етап 3. Для кожної змістовної МО складається список керуючих сигналів із множини Y, який забезпечує її (МО) виконання; визначаються тривалості кожного керуючого сигналу (у числі тактів) і періоди тактуючих сигналів автомата.

Етап 4. Кожній змістовній ЛУ ставиться у відповідність інформаційний сигнал із набору X.

Етап 5. З функціональних МПР $M_1, ..., M_G$ визначаються множини слів $S = \{s_1, ..., s_N\}$, МО $Y = \{y_1, ..., y_M\}$ і ЛУ $X = \{x_1, ..., x_L\}$, необхідні для реалізації операцій із набору $F = \{f_1, ..., f_G\}$. Множини S, Y, X визначаються як об'єднання, відповідно, множин слів S_g, МО Y_g, ЛУ X_g, що вводяться однією M_g, $g = 1, ..., G$.

Етап 6. Будуються закодовані ГСА $\Gamma_1, ..., \Gamma_G$, що відповідають змістовним МПР $M_1, ..., M_G$.

Етап 7. Закодовані ГСА $\Gamma_1, ..., \Gamma_G$ об'єднуються в одну ГСА Г. Задача об'єднання ГСА формулюється так. Припустимо, що у кожній з ГСА оператори не повторюються, але в різних ГСА можуть зустрічатися однакові оператори. Потрібно побудувати об'єднану ГСА Г, яка за умови, що повинна виконуватися операція f_g є рівносильною ГСА Γ_g ($g = 1, ..., G$).

Процедуру об'єднання ГСА можна умовно розбити на такі етапи.

▪ Для кожної граф-схеми Γ_g будується відповідна їй матрична схема алгоритму (МСА) C_g ($g = 1, ..., G$). Нехай ГСА Γ_g має початкову Y_{g0} і кінцеву Y_{gk} вершини, а також T_g інших операторних вершин з записаними у них різними операторами $Y_{g1}, ..., Y_{gT_g}$. Тоді МСА C_g, що відповідає ГСА Γ_g, являє собою квадратну матрицю, рядки якої позначено символами Y_{g0}, Y_{g1},... ..., Y_{gT_g}, а стовпці — символами $Y_{g1}, ..., Y_{gT_g}, Y_{gk}$. У цій матриці на перетині рядка Y_{gi} і стовпця Y_{gj} стоїть функція переходу α_{ij} від оператора Y_{gi} до оператора Y_{gj}. Перехід від ГСА Γ_g до МСА C_g ($g = 1, ..., G$) наступний: по ГСА Γ_g необхідно знайти усі функції переходу і занести їх у відповідні клітини матриці. Перехід від МСА C_g до ГСА Γ_g полягає у послідовному отриманні системи формул переходу, системи факторних (дужкових) формул переходу і після цього — ГСА Γ_g ($g = 1, ..., G$).

▪ Кожна МСА C_g кодується вектором ($e_{g1}, ..., e_{gh}$) значень змінних $p_1, ..., p_h$, де $p_i \notin \bigcup_{g=1}^{G} X_g$, а X_g — множина ЛУ граф-схеми Γ_g; $e_{gi} \in \{0, 1\}$; $i = 1, ..., h$; $2^h \geq G$.

▪ Кожній МСА C_g ставиться у відповідність кон'юнкція $P_g = p_1^{\#} ... p_h^{\#}$, де $p_i^{\#} = p_i$, якщо $e_{gi} = 1$, і $p_i^{\#} = p_i$, якщо $e_{gi} = 0$; $i = 1, ..., h$; ($e_{g1}, ..., e_{gh}$) — код МСА C_g.

▪ Будується об'єднана МСА C, рядки і стовпці якої позначено усіма операторами, що входять в об'єднання множин операторів МСА

$C1$, ..., C_G. Серед рядків, як і раніше, немає Y_k, а серед стовпців — Y_0. Елемент α_{ij} МСА C дорівнює $\alpha_{ij}^1 P_1 \vee ... \vee \alpha_{ij}^G P_G$, де α_{ij}^G — елемент МСА C_g, що стоїть на перетині рядка Y_{gi} і стовпця Y_{gj}.

Оскільки кон'юнкція P_g ($g = 1, ..., G$) дорівнює одиниці весь час, поки МСА C "працює" як МСА C_g, жоден оператор не змінює значень змінних p_1, ..., p_h (щодо цих змінних маємо порожній розподіл зсувів [19]), у зв'язку з чим МСА C можна мінімізувати з урахуванням розподілу зсувів.

Для переходу від мінімізованої МСА $C_{\min}$ до об'єднаної ГСА необхідно розбити МСА $C_{\min}$ на підматриці і скористатися методом побудови мінімальної ГСА або ж спробувати знайти близьку до мінімальної систему факторних формул переходу для кожної із підматриць. При цьому в обох випадках слід враховувати невизначеності двох типів, які виникають при об'єднанні часткових МПР [19]:

- Якщо число G МПР, що об'єднуються, строго менше 2^h, тобто не всі набори значень змінних p^1, ..., p^h використовуються для кодування МСА C_1, ..., C_G, усі формули переходу (або відмічені покриття формул переходу) є невизначеними на тих наборах значень змінних p_1, ..., p_h, x_1, ..., x_L, які покриваються кубами ($e_{g1}...e_{gh}x...x$), де ($e_{g1}...e_{gh}$) — невикористаний набір значень змінних p_1, ..., p_h.
- Якщо оператор Y_t не зустрічається в якихось МСА C_g ($g = 1, ..., G$), то формула переходу для Y_t (зазначене покриття Y_t) не визначена на тих наборах значень змінних, які покриваються кубом ($e_{g1}...e_{gh}x...$), де ($e_{g1}...e_{gh}$) — код МСА C_g.

Якщо тепер для підматриць побудувати відповідні ГСА, то після об'єднання однакових вихідних і вхідних операторних вершин в одну вершину прийдемо до необхідної об'єднаної ГСА Г.

Алгоритм синтезу керуючого автомата як автомата Мілі. В основі процедури синтезу КА лежить так званий канонічний метод структурного синтезу автоматів з пам'яттю, теоретичним фундаментом якого є теорема В.М. Глушкова про структурну повноту [1]. Вхідними даними для початку роботи методу служить абстрактний (цифровий) автомат з пам'яттю, заданий своїми таблицями переходів і виходів, а результатом роботи — рівняння булевих функцій переходів, виходів і функцій збудження елементів пам'яті в канонічній (операторній) формі подання. Якщо етапи 1–7 алгоритму синтезу КА є спільними для будь-яких типів КА, то надалі алгоритм синтезу розгалужується в залежності від типу КА, що синтезується.

Етап 8. Обирається тип КА. КА може бути синтезовано або як автомат Мілі, або як автомат Мура. Припустимо, що ми обрали автомат Мілі.

Етап 9. Здійснюється позначення станів у закодованій (об'єднаній) ГСА Г: символом a_1 позначається вхід вершини (логічної чи операторної), що слідує за початковою, а також вхід кінцевої вершини; входи усіх вершин, що слідують за операторними, позначаються різними символами a_j. Крім визначених станів може виникнути необхідність уведення додаткових станів, зокрема, для забезпечення стійкості роботи ОП (див. далі);

Етап 10. Закодована ГСА з позначками станів інтерпретується як автомат Мілі. З цією метою за даною ГСА будується граф переходів і виходів автомата Мілі. Число вершин у цьому графі дорівнює числу позначок станів a_j на ГСА. Кожному переходу автомата Мілі з одного стану в інший відповідає дуга графа. Дузі приписується набір ЛУ, що викликає цей перехід, а також набір керуючих сигналів, що відповідає даному переходу.

Етап 11. Здійснюється кодування станів КА. Довжина m двійкового кортежу, яка дозволяє однозначно закодувати стани КА, визначається з умови $2^m \geq M$, де M — число станів КА (бажано, щоб m було мінімальним). Число m визначає кількість тригерів, необхідних для організації пам'яті КА. У результаті кожному стану a_j відповідає одна визначена комбінація значень Q_1, ..., Q_m виходів тригерів.

Зазначимо, що спосіб кодування впливає на правильність формування керуючих сигналів і складність КА. Зокрема, неоптимальне кодування станів зумовлює появу негативних явищ — "гонок" сигналів та "проскоків" станів в КА.

Для усунення цього недоліку потрібно, зокрема, використовувати "сусіднє" кодування станів (код Грея). В КА, що не допускають сусіднього кодування, необхідно вводити додаткові стани.

Етап 12. Будується структурна таблиця переходів і виходів КА, а також функцій збудження елементів пам'яті (припускається, що нам відомі типи логічних елементів і типи тригерів, які будуть використовуватися у процесі синтезу ОП). Структурна таблиця КА будується за його графом переходів і виходів. У кожен рядок таблиці записують код поточного стану (у момент часу S), код стану переходу (у момент часу S+1), значення ЛУ, що забезпечують перехід, відповідні значення керуючих сигналів і функцій збудження тригерів.

Етап 13. Структурна таблиця інтерпретується як таблиця істинності для функцій збудження елементів пам'яті та функцій виходів (керуючих сигналів) як функцій кодів станів КА в момент часу S і ЛУ, що спричиняють перехід у стан КА у момент часу S+1. На підставі такої інтерпретації знаходяться аналітичні представлення названих функцій у вигляді МДНФ.

Етап 14. Знаходиться операторна форма представлення знайдених функцій збудження тригерів і керуючих сигналів.

Етап 15. Операторні форми розглядаються як форми для побудови структурної схеми КА і ескіза друкованої плати. На цьому процедура синтезу КА як автомата Мілі завершується.

Алгоритм синтезу керуючого автомата як автомата Мура.

Етап 8. Обирається тип КА. Припустимо, що ми обрали автомат Мура.

Етап 9. Здійснюється позначення станів закодованої ГСА для автомата Мура: символом a_1 позначаються початкова і кінцева вершини, а всі інші операторні вершини позначаються різними символами a_j (може виникнути необхідність уведення додаткових станів).

Етап 10. На графі переходів і виходів автомата Мура у вершинах графа, число яких дорівнює кількості позначок у закодованій ГСА, записують стани КА і відповідні їм керуючі сигнали, а дугам, що з'єднують вершини графа, приписують лише набори ЛУ, що спричиняють відповідні (вершинам графа) переходи КА зі стану в стан.

Етап 11. Кодування станів автомата Мура можна виконувати так само, як і для автомата Мілі. Однак при відповідному кодуванні керуючі сигнали можна знімати і безпосередньо з виходів тригерів автомата Мура, тобто КС для формування функцій y_j не потрібні.

Етапи 12–15 синтезу автоматів Мура майже повністю (з урахуванням особливостей цих автоматів) повторюють відповідні етапи синтезу автоматів Мілі.

Основні структури операційних автоматів. До основних структур ОА відносяться ОА з канонічною структурою, *I*-автомати, *M*-автомати, *IM*-автомати з паралельними та послідовними комбінаційними частинами (КЧ). Для синтезу структури ОА не підходять методи, що використовуються при синтезі КА, оскільки в основі цих методів лежать результати абстрактної теорії автоматів, а число різних станів, в яких може перебувати ОА, дуже велике. Структури ОА в заданому структурному базисі, який утворюють шини, регістри та КС, можна синтезувати безпосередньо за функцією, реалізація якої покладається на ОА. Як відомо, функція ОА задається множинами:

- слів $S = \{s_1, ..., s_N\}$, які можуть бути вхідними, вихідними і внутрішніми;
- МО $Y = \{y_m\} = \{s_\alpha := \varphi_m(s_{\beta 1}, ..., s_{\beta k})\}, m = 1, ..., M$ і $s_\alpha, s_{\beta 1}, ..., s_{\beta k} \in S$;
- ЛУ $X = \{x_l = \psi_l(s_{\gamma 1}, ..., s_{\gamma t})\}$, де $l = 1, ..., L$ і $s_{\gamma 1}, ..., s_{\gamma t} \in S$.

При цьому функцію ОА можна "витягти" безпосередньо зі змістовної (об'єднаної) графсхеми МПР, яку реалізує ОП.

Канонічна структура операційного автомата та її синтез. Алгоритм синтезу ОА з такою структурою:

- Словам $s_1,...,s_N$, які мають в МПР тип L, ставляться у відповідність регістри s_1, ..., s_N з довжинами $n_1,...,n_N$, що дорівнюють довжинам слів.
- Словам $s_{d1},...,s_{dH}$, які мають тип I, ставляться у відповідність вхідні полюси (входи) $d_1,...,d_H$ структурної схеми. Кожен вхід d_1, ..., d_H

з'єднується з відповідним регістром $s_{d1},...,\ s_{dH}$ шиною, що виходить із входу.

▪ Словам $s_{r1},...,s_{rQ}$, які мають тип O, ставляться у відповідність вихідні полюси (виходи) $r_1,\ ...,\ r_Q$ структурної схеми. Кожен регістр $s_{r1},...,s_{rQ}$ з'єднується з відповідним виходом $r_1, ,...,r_Q$ шиною, що виходить із регістра.

▪ Кожній МО $y_m \in Y$, що описується оператором присвоювання $s_\alpha := \varphi_m(s_{\beta 1},...,s_{\beta k})$, ставиться у відповідність КС φ_m, входи якої підключаються до регістрів $s_{\beta 1},...,s_{\beta k}$, а вихід з'єднується керованою шиною з регістром s_α. Керована шина позначається сигналом y_m, який ініціює МО присвоювання слову s_α значення $\varphi_m(s_{\beta 1},...,s_{\beta k})$. Для виконання МО передачі $s_\alpha := s_\beta$ й установки $s_\alpha :=$ const не потрібні КС, що обчислюють значення двійкового виразу.

▪ Кожній ЛУ $x_l = \psi_l(s_{\gamma 1},....,s_{\gamma t}),\ x_l \in X$, ставиться у відповідність КС ψ_l, входи якої з'єднуються з регістрами $s_{\gamma 1},...,s_{\gamma t}$, а вихід позначається сигналом x_l. Якщо ψ_l — тривіальна БФ, то ЛУ інтерпретується ланцюгом, що виходить з певного розряду регістра s_β.

Структуру ОА, яку отримують шляхом заміни кожного елемента функції ОА (слова, МО, ЛУ) відповідними елементами структурного базису, називають канонічною структурою.

Як вже зазначалося, в структурі ОА можна виділити три частини: пам'ять S, КС Ф і Ψ. Множину МО Y можна розділити на N підмножин $Y_1 = \{s_1 := \varphi_{mi}(S)\},....,YN = \{sN := \varphi_{mq}(S)\}$, кожна з яких складається із сукупності МО, що обчислюють значення лише одного слова із сукупності $S = \{s_1,...,s_N\}$. Різні множини Y_i та Y_j не містять спільних МО і тому КС Ф можна розділити на незалежні підсхеми $\Phi_1,...,\Phi_N$, які реалізують підмножини МО $Y_1,...,Y_N$. Підсхеми $\Phi_1,...,\Phi_N$ обслуговують відповідні регістри $s_1,...,s_N$ й у кожному такті можуть реалізувати по одній МО $y_{mi} \in Y_1,...,y_{mq} \in Y_N$. Аналогічно, КС Ψ, що обчислює значення ЛУ із множини X, може бути розділена на підсхеми $\Psi_1,...,\Psi_N$, що обчислюють значення із підмножин ЛУ $X_1,...,X_N$. Схему, що складається з регістра s_n і КС Φ_n і Ψ_n (n=1,...,N), можна розглядати як елементарний ОА, який називають операційним елементом. Отже, у загальному випадку ОА можна розглядати як сукупність операційних елементів $E_1,...,E_N$, число яких дорівнює кількості внутрішніх слів, що оброблюються МПР.

Властивості канонічних структур операційного автомата. Оскільки канонічна структура не вносить обмежень на сумісність МО (тобто усі функціонально сумісні МО можуть виконуватися паралельно в одному такті), то канонічна структура має максимальну продуктивність, яка може досягати N МО за такт, де N — кількість в МПР слів типу L. А тому витрати часу на виконання фіксованої МПР в ОА з канонічною структурою мінімальні у порівнянні з іншими варіантами структур. Менших витрат часу можна досягти, якщо змінити алгоритм виконання операцій в ОП, зокрема, шляхом аналітичних перетворень (див. далі).

Можна стверджувати, що канонічна структура має найвищу швидкодію (найменшу тривалість такту τ_{OA}) у порівнянні з іншими варіантами структур (хоча швидкодія різних структур ОА, що побудовані на одній і тій же елементній базі, розрізняється незначно).

У більшості випадків канонічна структура не є мінімальною за кількістю устаткування ("заліза"), що може використовуватись при реалізації фіксованого набору операцій. Це викликано рядом причин, для усунення яких розроблено різні формальні методи. Одна з причин — пам'ять ОА може бути надлишковою по відношенню до певного алгоритму. Як ми вже знаємо, для мінімізації пам'яті ОА існують методи, які дозволяють перетворити алгоритм таким чином, щоб мінімізувати в ньому кількість слів для подання даних. Крім цього, надмірність устаткування в КС із $Z = \{\varphi_1,...,\varphi_M, \psi_0,...,\psi_L\}$ часто спричинена тим, що декілька КС, наприклад $\varphi_a,...,\varphi_\omega$, можуть реалізовувати одну й ту саму функцію. Для зменшення кількості устаткування в ОА у цьому випадку можна, зокрема, замінити КС $\varphi_a,...,\varphi_\omega$ однією схемою. На цій ідеї засновано, зокрема, клас I-автоматів (див. далі).

Нарешті, набору КС Z може відповідати інший, еквівалентний за своїми функціями, набір КС, який породжує менші витрати устаткування. І в цьому випадку можна шляхом глибоких

перетворень алгоритму, що приводять до зміни набору МО і ЛУ, створити новий набір, для реалізації якого будуть потрібні КС Z із меншими витратами устаткування порівняно з КС Z. Але, як зазначалося в [3–6], техніка формальних перетворень алгоритмів з метою зміни часу їх реалізації й зменшення витрат устаткування носить достатньо складний і громіздкий характер, який, до того ж, важко формалізувати, і тому не дивно, що сьогодні такі перетворення виробляються, в основному, "вручну" і, до того ж, евристичними методами.

Отже, канонічна структура, що реалізує задану функціональну МПР, має максимально можливу для даної МПР продуктивність і максимальну швидкодію. Оскільки канонічна структура синтезується безпосередньо за функціональною МПР без використання жодних процедур мінімізації витрат на "залізо", то така структура, в загальному випадку, є надлишковою за кількістю устаткування, що використовується в ній.

I-автомати та їх синтез. До класу *I*-автоматів належать ОА зі структурою, у якій мінімально можлива кількість КС забезпечує можливість одночасного виконання усіх функціонально сумісних МО. Уведемо необхідні поняття. Двійкові вирази $C_{\alpha 1}{*}C_{\alpha 2}{*}...{*}C_{\alpha p}$ і $C_{\beta 1}{*}C_{\beta 2}{*}...{*}C_{\beta q}$ (тут C_α, C_β — аргументи оператора, що являють собою слова, їхні інверсії і константи; * — знаки двійкових операцій), що стоять у правій частині МО (оператора присвоювання), називаються еквівалентними, якщо один з двійкових виразів може бути отриманий з іншого шляхом [18]:

- заміні слова C_α словом C_β або $\neg C_\beta$;
- заміні слова C_α константою (зокрема, нулем) і навпаки;
- заміні одних констант іншими, у тому числі і нульовими;
- рівносильними перетвореннями виразу $C_{\alpha 1}{*}C_{\alpha 2}{*}...{*}C_{\alpha p}$.

Еквівалентними МО називаються МО із еквівалентними двійковими виразами. З цього визначення, зокрема, випливає, що МО будуть еквівалентними, якщо функції в їх операторах мають однакові імена. Так, $s_{\alpha 1} := \varphi_m(s_{\alpha 2}, ..., s_{\alpha p})$ і $s_{\beta 1} := \varphi_m(s_{\beta 2}, ..., s_{\beta q})$ — еквівалентні МО. При цьому, у загальному випадку, $p \neq q$, тобто кількість аргументів p і q у функціях φ_m може бути різною.

Зазначимо, еквівалентність МО означає, що для обчислення двійкових виразів, які відповідають цим МО, може використовуватися одна КС. Але це виключає сумісність цих МО через структурні обмеження, що виникають при цьому, і, у загальному випадку, призводить до збільшення часу виконання операцій.

Для побудови структури, що реалізує сукупність еквівалентних МО y_a, ...,y_w, уводиться спеціальна форма подання таких МО — узагальнений оператор, який має вигляд: $s_\gamma := A_{\alpha 1}{*}A_{\alpha 2}{*}...{*}A_{\alpha r}$, де $A_{\alpha 1}, A_{\alpha 2}, ..., A_{\alpha r}$ — допоміжні змінні, що приймають певні значення при виконанні МО $y_a, ..., y_w$. Оскільки МО $y_a, ..., y_w$ несумісні, то довільна допоміжна змінна визначається завжди однозначно. Алгоритм синтезу *I*-автомата, по суті, зводиться до перетворення заданого набору МО Y в сукупність узагальнених операторів, які виступають як форми для побудови структурної схеми *I*-автомата. Алгоритм полягає в наступному :

- Множина МО Y розбивається на підмножини $Y_1, ..., Y_N$, що відповідають внутрішнім словам (регістрам) $s_1, ..., s_N$. При цьому МО $s_\alpha := \varphi_m(s_{\beta 1}, ..., s_{\beta k})$ приписується множині Y_α.
- На підмножинах Y_n, $n = 1, ..., N$ виділяються класи еквівалентних МО K_{nj}, $j = 1, ..., J_n$.
- Для класу K_{nj}, що містить не менше двох МО, будується узагальнений оператор. Якщо клас K_{nj} містить лише одну МО, то узагальненим оператором для нього є сама МО.
- Виходячи з опису слів, списку узагальнених операторів і ЛУ будується структурна схема *I*-автомата.

Властивості I-автоматів. Легко бачити, що алгоритм синтезу *I*-автоматів базується на можливості подання структури ОА у вигляді композиції операційних елементів $E_1, ..., E_N$, де N — число слів типу L в МПР. Оскільки в структурі *I*-автомата, що виникає у результаті застосування даного алгоритму синтезу, кожен операційний елемент E_n використовується для обчисленням значень лише одного слова s_n

($n = 1, ..., N$), то така структура і не вносить обмежень на сумісність МО, і мінімізує витрати устаткування в ОА. Наслідок цього — максимальна продуктивність, яка при наявності N операційних елементів може, потенційно, досягати N МО за такт, і мінімальне число КС, які використовуються в ОА для виконання МО. Отже, у порівнянні з ОА з канонічною структурою продуктивність I-автоматів не нижче, а витрати устаткування — менше.

М-автомати та їх синтез. У структурі I-автомата можуть міститися еквівалентні за своїми функціями КС, які використовуються для обслуговування різних регістрів. При заданій МПР витрати устаткування в ОА можна мінімізувати, якщо кожну КС φ_m використовувати для виконання всіх еквівалентних МО з множини Y. ОА, синтезовані на основі даного принципу, називають M-автоматами.

Для обчислення будь-якого двійкового виразу $\varphi_k(s_{\beta 1}, ..., s_{\beta k})$ у структурі M-автомата створюється одна КС Ф, рівнодоступна по відношенню до всіх регістрів $s_1, ..., s_N$, що оперують зі словами типу L. Для вибірки слів із регістрів $s_1, ..., s_N$ на шини $A1$ і $A2$, по яким операнди, що беруть участь в МО, надходять на вхід КС Ф, використовуються керуючі сигнали $a_1, ..., a_N$ та $b_1, ..., b_N$ відповідно: сигнал a_i ініціює передачу $A_1 := s_i$, а сигнал b_j — передачу $A2 := s_j$. КС Ф налаштовується на виконання перетворення $\varphi_k(A_1, A_2) = R$ керуючим сигналом φ_k ($k = 1, ..., K$). Завантаження обчисленого значення R в будь-який регістр s_n ініціюється керуючим сигналом d_n ($n = 1, ..., N$).

Отже, у процесі роботи M-автомат генерує специфічний набір керуючих сигналів $\{a_i\}$, $\{b_j\}$, $\{\varphi_k\}$, $\{d_n\}$, які ініціюють виконання МО $s_n := \varphi k(s_i, s_j)$. У результаті виникає новий набір МО $\{A_1 := s_i\} \cup \{A_2 := s_j\} \cup \{R := \varphi_k(A_1, A_2)\} \cup \cup \{s_n := R\}$, достатній для реалізації будь-якої МО функціональної МПР (зазначимо, існують МО, наприклад унарні чи установки типу s_n:=const, при виконанні яких деякі із сигналів a_i, b_j, φ_k, d_n не виробляються).

Алгоритм синтезу M-автоматів зводиться до породження на основі списку МО Y сукупності операторів, притаманних структурі M-автомата (вибірки слів на шини, перетворення слів і завантаження результату в регістри), і складається з наступних етапів.

Етап 1. Здійснюється розподіл регістрів на шини A_1 і A_2 з мінімізацією числа керованих шин, що використовуються для передачі операндів на входи КС Ф.

Етап 2. Визначаються формати і значення слів A_1 , A_2. Як правило, приймається наступна угода про порядок подання внутрішніх слів $s_1, ..., s_N$ допоміжними словами A_1, A_2: молодші розряди слів $s_{\alpha 1}, ..., s_{\alpha p}$, що складають множину A_i ($i = 1, 2$), співставляються з молодшим розрядом слова A_i. Кількість розрядів у слові A_i визначається максимальним числом розрядів в словах $s_{\alpha 1}, ..., s_{\alpha p}$, включених в множину A_i.

Етап 3. Визначаються оператори, що реалізуються M-автоматом (тут, на відміну від множини МО Y із МПР, під операторами розуміються МО, притаманні структурі M-автомата), і здійснюється кодування МО наборами керуючих сигналів.

Етап 4. Визначаються класи еквівалентних МО, будуються узагальнені оператори і структурна схема M-автомата.

Етап 5. Оскільки у кожному такті M-автомат реалізує лише одну МО, яка ініціюється набором сигналів (a_i, b_j, φ_k, d_n), то закодована ГСА, яка визначає функцію КА, повинна бути перетворена наступним чином: кожна операторна вершина, яка містить q функціонально сумісних МО $y_{k1}, ..., y_{kq}$, замінюється послідовністю q операторних вершин, що містять по одній МО $y_{k1}, ..., y_{kq}$; кожен символ $y_k \in Y$ замінюється відповідним набором (a_i, b_j, φ_k, d_n).

Властивості I-автоматів. Продуктивність M-автомата має мінімальне значення, яке дорівнює одній МО за такт. У M-автоматі, у порівнянні з I-автоматом, швидкодія відрізняється вкрай незначно, а витрати устаткування менші, оскільки кожна КС $\varphi_k \in \Phi$ використовується для виконання всіх еквівалентних МО з множини Y.

IM-автомати та їх синтез. Класи I- та M-автоматів володіють діаметрально протилежними властивостями. Слід очікувати, що між цими двома класами ОА існують ОА з про-

міжними властивостями. Один з варіантів таких ОА утворює клас *IM*-автоматів, які мають досить високу продуктивність при помірних витратах устаткування. Для виконання МО в *IM*-автоматах можуть використовуватись паралельні й послідовні КС, що породжують структури, які називають, відповідно, *IM*-автоматами з паралельною і послідовною КЧ.

IM-автомат з паралельною КЧ можна розглядати як композицію з B ($1<B<N$) M-автоматів, що мають спільну пам'ять s_1, ..., s_N. Тому синтез *IM*-автоматів з паралельною КЧ зводиться до розбиття множини МО Y на B підмножин Y_1, ..., Y_B і синтезу BM-автоматів, які реалізують зазначені підмножини МО. В одному такті автомат може виконувати до B МО включно, що ініціюються набором керуючих сигналів, кількість яких залежить як від величини B, так і характеру МО, і тому може змінюватись у досить широких межах. Величина B визначається вимогами до швидкодії ОП, насамперед, обмеженням на час виконання операцій і затратами устаткування.

Принцип послідовної організації КЧ ОА призводить до структур ОА, у яких, для прикладу, КЧ складається з трьох КС Φ_1, Φ_2, Φ_3, що реалізують операції з множин $\{f_k\}$, $\{g_l\}$, $\{h_m\}$ відповідно (при використанні таких ОА як ОА процесорів КС Φ_1, Φ_2, Φ_3, зазвичай, реалізують функції формувача, суматора та зсувача кодів відповідно [18]). Для нашого прикладу ОА за один такт реалізує перетворення $s_n := h_m(g_l(s_i, f_k(s_j)))$, еквівалентне трьом послідовно виконуваним МО f_k, g_l, h_m над словами s_i, s_j з метою обчислення слова s_n. Це перетворення ініціюється набором сигналів (a_i, b_j, f_k, g_l, h_m, d_n). При цьому під впливом сигналів a_i і b_j, які ініціюють передачі $A_1 := s_i$ і $A_2 := s_j$, слова s_i і s_j "вибираються" на входи A_1 і A_2; сигнали f_k, g_l, h_m, які ініціюють перетворення: $A_3 := f_k(A_2)$, $A_4 := g_l(A_1, A_3)$, $R := h_m(A_4)$, де A_3, A_4, R — допоміжні змінні, "налаштовують" КС Φ_1, Φ_2, Φ_3 на виконання необхідних МО; сигнал d_n ініціює запис $s_n := R$ результату R в пам'ять ОА. За рахунок цього максимальна продуктивність *IM*-автомата з даною структурою в три рази перевищує продуктивність M-автомата.

Синтез *IM*-автомата з послідовною КЧ здійснюється на основі МПР шляхом подання послідовностей МО $y_{\alpha 1}$, ..., $y_{\alpha r} \in Y$ у формі виразів $s_n := h_m(g_l(s_i, f_k(s_j)))$. Для цього МПР ділиться на лінійні ділянки (ЛД), що складаються з послідовності операторів O_1, ..., O_r. ЛД класифікуються за рангами $r = 1, ..., T$. До рангу r відноситься ЛД, що складається з r операторів O_1, ..., O_r. В результаті функція ОА подається множинами L_1, ..., L_R ЛД.

Потім визначається множина виразів, породжуваних цими ЛД. ЛД першого рангу L_1 відповідають МО y_{a1}, ..., $y_{ar} \in Y$. Вирази, що належать до ЛД $O_{\gamma 1}$, ..., $O_{\gamma r}$ рангу $r = 2, ..., T$, знаходяться в такий спосіб. Серед МО, що входять до складу операторів ЛД, виділяються МО y_{b1}, ..., y_{bt}, пов'язані з обчисленням одного і того ж слова s_n. Шляхом послідовних підстановок виразів $y_{b1} \to y_{b2} \to ... \to y_{bt}$ формується МО $s_n := \varphi_m(s_{in}, ..., s_{im})$, яка еквівалентна послідовності МО $y_{b1}, y_{b2}, ..., y_{bt}$. МО $y_{b1}, y_{b2}, ..., y_{bt}$ виключаються з ЛД, і процес породження виразів виду $s_n := h_m(g_l(s_i, f_k(s_j)))$ повторюється для решти МО лінійної дільники.

Множина виразів M_r, що породжуються ЛД L_r рангу r, визначається шляхом об'єднання виразів M_{ir} (i=1, ..., r), що породжуються окремими ЛД. Множина виразів M, що породжуються МПР, визначається шляхом об'єднання виразів M_r (r=1, ..., T). Множину M можна розглядати як множину МО, реалізація яких покладається на ОА. Застосуванням алгоритму синтезу M-автоматів до множини МО M визначається набір операторів $A_1 := s_i$, $A_2 := s_j$, $A3 := f_k(A_2)$, $A_4 := g_l(A_1, A_3)$, $R := h_m(A_4)$, $s_n := R$, що реалізуються схемами вибірки операндів A_1, A_2, КЧ і схемою запису результату R в регістри $s1$, ..., s_N. За набором операторів будується структурна схема *IM*-автомата. Кількість рівнів в КЧ ОА визначається максимальним числом дій в двійкових виразах M.

Властивості IM-автоматів. Структурна організація *IM*-автоматів з паралельною КЧ добре пристосована для реалізації МПР, у яких велике число операторів містять декілька сумісних МО, а лінійні ділянки не містять МО, пов'язані з обчисленням одного слова. Така

структура вносить деякі обмеження на сумісність МО і, разом з цим, забезпечує виконання за такт більше однієї МО функціональної МПР. Максимальна продуктивність *IM*-автомата з *B* паралельними КС ($1<B<N$), дорівнює *B* МО за такт і зростає зі збільшенням числа КС.

IM-автомати з послідовною КЧ доцільно застосовувати для реалізації МПР, у яких присутні багаторазово виконувані послідовності МО, що забезпечують обчислення одного слова $s_n \in S$. Такі послідовності найбільш легко виявляються на лінійних ділянках МПР.

Висновки

У першій частині роботи здійснюється аналіз задачі проектування ОП, деяких основних методів її вирішення, проблем, що виникають при цьому, і підходів до їх подолання. Аналіз стосується, насамперед, функцій, реалізація яких покладається на ОП, і лише частково, на змістовному рівні (неформально), торкається як форми, у якій ці функції задаються, так і техніки виконання їх (цих функцій) оптимізуючих перетворень.

З урахуванням застережень, здійснених на етапі аналізу, формулюється процедура визначення структури (архітектури) ОП і порядку його функціонування, яка базується на принципі мікропрограмного керування. Виявляється, точка зору на процес функціонування ОП як процес реалізації принципу мікропрограмного керування є ефективною (результативною) не лише з практичної точки зору, але й теоретичної, бо дозволяє формалізувати процес проектування ОП.

Цю формалізацію здійснив В.М.Глушков, який як узагальнену модель ОП запропонував використовувати поняття дискретного перетворювача інформації (композиції зі зворотним зв'язком двох автоматів – керуючого і операційного, яким відповідають, як правило, автомати Мілі і Мура), а як формальні засоби, що описують "програми" обробки даних в ОП, – апарат систем алгоритмічних алгебр. Зазначимо, даний підхід носить загальний характер і може бути використаний не лише в моделях однопроцесорних обчислювальних систем, але й у тих випадках, коли ОП є багатопроцесорним комплексом. У другій частині роботи буде показано, що цей підхід дозволяє не лише з єдиних позицій підійти до вирішення багатоаспектних проблем, виявлених у процесі аналізу задачі проектування ОП, але й автоматизувати процедуру синтезу ОП потрібної якості на базі запропонованого варіанту (одного із можливих) інструментального засобу.

ЛІТЕРАТУРА

1. *Глушков В.М.* Синтез цифровых автоматов. М.: Физматгиз., 1962. 476 с.
2. *Глушков В.М.* Теория автоматов и вопросы проектирования структур цифровых машин. Кибернетика. 1965. № 1. С. 3–11.
3. *Глушков В.М.* Теория автоматов и формальные преобразования микропрограмм. Кибернетика. 1965. №5. С. 1–10.
4. *Глушков В.М., Летичевский А.А.* Теория дискретных преобразователей. Избранные вопросы алгебры и логики. Новосибирск: Наука, 1973. С. 5–40.
5. *Капитонова Ю.В., Летичевский А.А.* Математическая теория проектирования вычислительных систем. М.: Наука, 1988. 295 с.
6. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. 3-е изд., перераб. и доп. Киев: Наук. думка, 1989. 376 с.
7. *Анисимов А.В.* Рекурсивные преобразователи информации. Киев: Вища шк., 1987. 231 с.
8. *Bauer F.L., Wossner H.* Algorithmische sprache und programmentwicklung. Berlin ets.: Springer Verlag, 1981. 513 p.
9. *Ершов А.П.* Трансформационная машина: тема и вариации Проблемы теоретического и системного программирования. Новосибирск. НГУ, 1979. С. 5–45.
10. *Петрушенко А.Н.* О диалоговых вычислениях в алгоритмических алгебрах. Кибернетика. 1990. №1. С. 13–20.
11. *Петрушенко А.Н.* Об одном подходе к проблеме автоматизации оптимизирующих преобразований алгоритмов и программ. Кибернетика и системный анализ. 1991. №5. С. 127–136.
12. *Петрушенко А.Н.* Об одном подходе к решению проблемы общения человека с вычислительной системой на естественном языке. Проблемы программирования: Сб. науч. трудов. вып. 3. 1998. С. 65–72.

13. *Петрушенко А.Н., Хохлов В.А.* Об использовании естественного языка для представления абстрактных типов данных и полиморфизма. Проблемы программирования. 1999. №1. С. 76–83.
14. *Петрушенко А.Н., Хохлов В.А., Ткачев В.А., Шепетухин Е.С.* Диалоговая трансформационная машина: некоторые функциональные возможности. Проблемы программирования. 2000. № 1–2 (Спец. выпуск). С. 323–334.
15. *Петрушенко А.М., Хохлов В.А.* Концепція діалогових обчислень та деякі проблеми автоматизації програмування. Проблемы программирования. 2004. № 2–3 (Спец. выпуск). С. 37–47.
16. *Самофалов К.Г., Корнейчук В.Н., Тарасенко В.П.* Цифровые ЭВМ. К.: Вища шк., 1989. 423 с.
17. *Самофалов К.Г., Корнейчук В.Н., Тарасенко В.П., Жабин В.Н.* Цифровые ЭВМ. Практикум. К.: Вища шк., 1990. 215с.
18. *Майоров С.А., Новиков Г.И.* Структура электронных вычислительных машин. Л.: Машиностроение, 1979. 384 с
19. *Баранов С.И.* Синтез микропрограммных автоматов. Л.: Энергия, 1979. 232с.
20. *Калужнин Л.А.* Об алгоритмизиции математических задач. Проблемы кибернетики. Вып. 2. М.: Физматгиз, 1959. С. 51–67.
21. *Ляпунов А. А.* О логических схемах программ. Проблемы кибернетики. Вып. 1, М.: Физматгиз, 1958. С. 46–74.
22. *Янов Ю.И.* О логических схемах алгоритмов. Проблемы кибернетики. Вып. 1. М.: Физматгиз, 1958. С. 75–127.
23. *Яблонский С.В.* Основные понятия кибернетики. Проблемы кибернетики. Вып. 2. М.: Физматгиз, 1959. С. 7–38.
24. *Ершов А.П.* Операторные алгоритмы. 3 (об операторных схемах Янова). Проблемы кибернетики. Вып. 20. М.: Физматгиз, 1968. С. 181–200.
25. *Системы компьютерной* алгебры семейства АНАЛИТИК. Теория. Реализация. Применение. К., 2010. 762 с.
26. *Разевиг В.Д.* Применение программ PCAD и PSpise для схемотехнического моделирования на ПЭВМ: В 4 выпусках. Вып. 1: Общие сведения. Графический ввод схем. М.: Радио и связь, 1992. 72 с.

Надійшла 16.10.2019

REFERENCES

1. *Glushkov, V.M.*, 1962. Sintez tsifrovykh avtomatov. M.: Fizmatgiz. 476 p. (In Russian).
2. Glushkov, V.M., 1965. "Teoriya avtomatov i voprosy proyektirovaniya struktur tsifrovykh mashin", Kibernetika, 1, pp. 3–11. (In Russian).
3. *Glushkov, V.M.*, 1965. "Teoriya avtomatov i formalnyye preobrazovaniya mikroprogramm". Kibernetika, 5, pp. 1–10. (In Russian).
4. *Glushkov, V.M., Letichevskiy, A.A.*, 1973. "Teoriya diskretnykh preobrazovateley". Izbrannyye voprosy algebry i logiki. Novosibirsk: Nauka, pp. 5–40. (In Russian).
5. *Kapitonova, Yu.V., Letichevskiy, A.A.*, 1988. Matematicheskaya teoriya proyektirovaniya vychislitelnykh sistem. M.: Nauka, 295 p. (In Russian).
6. *Glushkov, V.M., Tseytlin, G.Ye., Yushchenko, Ye.L.*, 1989. Yazyki. Programmirovaniye. 3-ye izd., perab. i dop. Kyiv: Nauk. dumka, 376 p. (In Russian).
7. *Anisimov, A.V.*, 1987. Rekursivnyye preobrazovateli informatsii. Kyiv: Vishcha shk., 231 p. (In Russian).
8. *Bauer, F.L., Wossner, H.*, 1981. Algorithmische sprache und programmentwicklung. Berlin: Springer Verlag, 513 p.
9. *Yershov, A.P.*, 1979. "Transformatsionnaya mashina: tema i variatsii". Problemy teoreticheskogo i sistemnogo programmirovaniya. Novosibirsk, NGU, pp. 5–45. (In Russian).
10. *Petrushenko, A.N.*, 1990. "O dialogovykh vychisleniyakh v algoritmicheskikh algebrakh". Kibernetika, 1, pp. 13–20. (In Russian).
11. *Petrushenko, A.N.*, 1991. "Ob odnom podkhode k probleme avtomatizatsii optimiziruyushchikh preobrazovaniy algoritmov i programm". Kibernetika i sistemnyy analiz, 5, pp. 127–136. (In Russian).
12. *Petrushenko, A.N.*, 1998. "Ob odnom podkhode k resheniyu problemy obshcheniya cheloveka s vychislitelnoy sistemoy na yestestvennom yazyke". Problemy programmirovaniya: Sb. nauch. trudov, 3, pp. 65–72. (In Russian).
13. *Petrushenko, A.N., Khokhlov, V.A.*, 1999. "Ob ispolzovanii yestestvennogo yazyka dlya predstavleniya abstraktnykh tipov dannykh i polimorfizma". Problemy programmirovaniya, 1, pp. 76–83. (In Russian).
14. P*etrushenko, A.N., Khokhlov, V.A., Tkachev, V.A., Shepetukhin, Ye.S.*, 2000. "Dialogovaya transformatsionnaya mashina: nekotoryye funktsionalnyye vozmozhnosti". Problemy programmirovaniya, 1–2 (Spets. vypusk), pp. 323–334. (In Russian).
15. Petrushenko A.M., Khokhlov V.A., 2004. Kontseptsiya dialohovykh obchyslen ta deyaki problemy avtomatyzatsiyi prohramuvannya. Problemy prohrammyrovanyya, 2–3 (Spets. vypusk), pp. 37–47. (In Ukrainian).
16. *Samofalov, K.G., Korneychuk, V.N., Tarasenko, V.P.*, 1989. Tsifrovyye EVM. K.: Vishcha shk., 423 p. (In Russian).
17. *Samofalov, K.G., Korneychuk, V.N., Tarasenko, V.P., Zhabin, V.N.*, 1990. Tsifrovyye EVM. Praktikum. K.: Vish. shk., 215 p. (In Russian).
18. *Mayorov, A., Novikov, G.I.*, 1979. Struktura elektronnykh vychislitelnykh mashin. L.: Mashinostroyeniye. 384 p. (In Russian).
19. *Baranov, S.I.,* 1979. Sintez mikroprogrammnykh avtomatov. L.: Energiya. 232 p. (In Russian).

20. *Kaluzhnin, L.A.,* 1959. "Ob algoritmizitsii matematicheskikh zadach". Problemy kibernetiki., 2, M.: Fizmatgiz, pp. 51–67. (In Russian).
21. *Lyapunov, A.A.*, 1958. "O logicheskikh skhemakh programm". Problemy kibernetiki., 1, M.: Fizmatgiz, pp. 46–74. (In Russian).
22. *Yanov, Yu.I.,* 1958. "O logicheskikh skhemakh algoritmov". Problemy kibernetiki. M.: Fizmatgiz, 1, pp. 75–127. (In Russian).
23. *Yablonski,y S.V.*, 1959. "Osnovnyye ponyatiya kibernetiki". Problemy kibernetiki. M.: Fizmatgiz, 1, pp. 7–38. (In Russian).
24. *Yershov, A.P.*, 1968. "Operatornyye algoritmy". 3 (ob operatornykh skhemakh Yanova). Problemy kibernetiki. M.: Fizmatgiz, 2, pp. 181–200. (In Russian).
25. *Sistemy kompyuternoy* algebry semeystva ANALITIK. Teoriya. Primeneniye. K., 2010. 762 p. (In Russian).
26. *Razevig, V.D.*, 1992. Primeneniye programm PCAD i PSpise dlya skhemotekhnicheskogo modelirovaniya na PEVM: V 4 vyp. 1: Obshchiye svedeniya. Graficheskiy vvod skhem. M.: Radio i svyaz. 72 p. (In Russian).

Received 16.10.2019

A.M. Petruchenko, PhD in Phys.-Math Sciences, Associate Professor,
Taras Shevchenko National University of Kyiv,
03022, Kyiv, Glushkov ave., 4D, Ukraine,
anatoly@mytaskhelper.com

THE PRINCIPLE OF FIRMWARE CONTROL AND DESIGN AUTOMATION OF OPERATING DEVICES. I

Introduction. Promising areas of research that are developing both in Ukraine and abroad include the so-called transformational synthesis methods. According to these methods, a computing system is obtained by phasing the initial description of the system (setting the task) according to the rules,which is knowledge about the problem being solved. In this area of research, two interrelated directions can be distinguished – the theoretical and applied. The theoretical direction requires, in particular, the presentation of various computational models that describe particular parts of computing systems, and the study of basic transformations in these models. The applied direction is associated with the creation of a transformation machine, the commands for which are basic transformations, and the data are expressions in the language over which these transformations are given. The Ukrainian Algebra-Cybernetic School researches the computational model, which is based on the concept of a discrete information converter. Representation of the computing process in this form allowed V.M. Glushkov and his students to create a new theoretical direction in the applied theory of algorithms – the algebra of algorithms. A characteristic feature of this direction is the commonality of mathematical models and methods for designing programmes and equipment. The apparatus of algebra of algorithms is the basis of the dialogue transformation machine – the main object of this article.

The purpose is to demonstrate the inextricable link between the fundamental concepts of the general theory of computer systems design and practical methods of designing software and hardware of computer technology, as well as new technological capabilities that arise when using the apparatus of algebra of algorithms in the process of designing programmes and equipment using an interactive transformational machine.

Methods. When implementing the tools (conversational transformation machine) and the synthesis algorithm of operating devices, we used the algebraic-grammatical method of representing knowledge, the method of constructing operating devices based on the principle of microprogram control, the methods of abstract and structural theory of automata, the methods of algebra of algorithms, etc.

Results. Synthesis methods for operating devices developed for the language of graph diagrams of algorithms and the language of logical diagrams of algorithms are extended to the language CAA\D – the input language of the dialogue transformation machine. Based on the dialogue transformational machine, a toolkit has been developed that embodies the V.M.Glushkov mathematical model and allows the complex automation of the operating devices: from setting the task to obtaining a sketch of the printed circuit board.

Conclusions. The integral algebraic-grammatical apparatus underlying the dialogue transformational machine combines algebraic, logical, and grammatical formalisms and is focused on the multi-level structural design of classes of algorithms and associated programmes (serial and parallel) and hardware. It is characterized by the analytical style of the specifications of programmes and equipment, focused on their optimizing transformations in order to achieve the necessary quality indicators. At the same time, using the CAA\Dv language as the input language of the dialog transformational machine allows you to increase the "intelligence" of the computer to a level that provides direct communication with it, in particular, inexperienced in programming users who are specialists in specific areas.

Keywords: *a principle of microprogram control, discrete information converter, operating device, algebra of algorithms, algebraic-grammatical method of knowledge representation, interactive transformation machine.*

А.Н. Петрушенко, кандидат физ.-мат. наук, доцент, Киевский национальный университет
имени Тараса Шевченко, 03022, г. Киев, просп. Академика Глушкова, 4Д,
факультет компьютерных наук и кибернетики,
anatoly@mytaskhelper.com

ПРИНЦИП МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ И АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ОПЕРАЦИОННЫХ УСТРОЙСТВ. I

Введение. Одной из определяющих тенденций развития науки в целом является ее математизация. Не являются исключением и науки, которые исследуют компьютеры, принципы их строения и функционирования. Создание компьютеров и универсальных алгоритмических языков приводит к изменению акцентов при оценке значения математизации — ее уровень является теперь не только одним из главных показателей уровня научности, системности представлений о тех или иных предметных областях, но и определяет возможности автоматизации процессов, протекающих в этих областях.

Цель статьи. Компьютеры изменяют не только отношение к математизации, но и ее характер, ее сущность. В частности, возник метод научных исследований – имитационное моделирование, который соединил в себе черты классических дедуктивных и экспериментальных методов, присущих математике и физике соответственно. Но такой метод проведения эксперимента еще не есть настоящая математизация — для ее достижения необходимы алгебры алгоритмов. Именно в существовании такой алгебры и состоит разница между настоящей математизацией и простой формализацией знаний. Разработка такого аппарата связана с работами В.М.Глушкова, который ввел понятие дискретного преобразователя информации и системы алгоритмических (их сначала называли микропрограммными) алгебр. По Глушкову, каждая модель математизации должна состоять: из области объекта, которая должна быть изучена формальными математическими методами; из формального языка, подходящего для более или менее адекватного описания этого объекта; формальной модели вывода, включающей как общие средства логического вывода, так и средства вывода, специально разработанные для этого языка (алгебру данного языка). Формальный язык вместе с формальной моделью вывода образуют формальную теорию. Чтобы эта теория стала настоящим дедуктивным аппаратом, она должна быть воплощена в инструменте вывода, который составляет четвертую часть модели математизации. В настоящее время известны два типа таких инструментов: человеческий мозг и компьютер. Формальная теория, усвоенная мозгом или компьютером, названа системой познания. Легко видеть, что реализация идей В.М.Глушкова в полном объеме сопряжена с корректировкой целей математизации – одной из главных целей которой является повышение интеллекта компьютера до уровня, который обеспечивает непосредственное общение с компьютером даже несведущих в программировании пользователей – специалистов в конкретных предметных областях. Цель данной статьи — демонстрация одного из возможных подходов к реализации инструментария, воплощающего модель математизации В.М. Глушкова и возможностей данного инструментария на примере синтеза операционных устройств.

Методы. При реализации инструментария и алгоритма синтеза операционных устройств использовался алгебро-грамматический метод представления знаний, метод построения операционных устройств на базе принципа микропрограммного управления, методы абстрактной и структурной теории автоматов, методы алгебры алгоритмов и т.д.

Результат. Разработан инструментарий, воплощающий модель математизации В.М. Глушкова, позволяющий осуществить комплексную автоматизацию проектирования операционных устройств.

Выводы. Необходима работа по приведению инструментария к товарному виду.

Ключевые слова: *принцип микропрограммного управления, дискретный преобразователь информации, операционное устройство, алгебра алгоритмов, алгебро-грамматический метод представления знаний, диалоговая трансформационная машина.*