УДК 004.94 : 004.2

Параметризація граф-схем алгоритмів цифрових пристроїв керування

Роман Маркович Бабаков
доцент, д-р техн. наук
ORCID: 0000-0001-7196-0912
r.babakov@donnu.edu.ua

Донецький національний університет імені Василя Стуса

Олександр Олександрович Баркалов
професор, д-р техн. наук
ORCID: 0000-0002-4791-2088
a.barkalov@imei.uz.zgora.pl

Університет Зеленогурський

Ключові слова:
граф-схема алгоритму;
цифрові пристрої керування;
оптимізація схеми;
параметри;
псевдовипадкова генерація.

Розглядається науково-практична задача визначення множини параметрів граф-схем алгоритмів у контексті подальшої псевдовипадкової генерації граф-схем для дослідження ефективності методів синтезу і оптимізації цифрових пристроїв керування. Розглянуто структурні компоненти та визначено загальні параметри граф-схем алгоритмів, які традиційно використовуються для опису алгоритмів роботи цифрових пристроїв керування. Проаналізовано основні класи цифрових пристроїв керування, як-от мікропрограмний автомат (автомат з жорсткою логікою), мікропрограмний пристрій керування (автомат з програмувальною логікою) та композиційний мікропрограмний пристрій керування. Для зазначених класів пристроїв розглянуто основні методи оптимізації апаратурних витрат, серед яких кодування наборів мікрооперацій, заміна вхідних змінних, операційне перетворення кодів станів тощо. Для кожного з розглянутих класів пристроїв керування та методів оптимізації запропоновані набори параметрів граф-схеми алгоритму, які впливають на ефективність застосування відповідних структур і методів та характеризують як функцію переходів, так і функцію виходів пристрою керування. Для окремих параметрів визначено допустимий діапазон змін та співвідношення або взаємовиключність з іншими параметрами граф-схем. Наведені ілюстративні приклади визначення окремих параметрів за заданою граф-схемою. Надано рекомендації щодо використання запропонованих параметрів для псевдовипадкової генерації граф-схем алгоритмів. Визначено такі загальні вимоги щодо коректної псевдовипадкової генерації граф-схем алгоритмів: можливість досягнення кінцевої вершини з будь-якої іншої вершини; відсутність вершин, у яких вхід не зв’язаний з виходом іншої вершини; відсутність повторення логічних умов у послідовно розташованих вершинах; наявність хоча б однієї операторної вершини тощо.

DOI: https://doi.org/10.31558/2786-9482.2024.2.1

Вступ

Цифрові системи займають важливе місце в діяльності людства [1, 2]. Функціонування і взаємодія окремих компонентів цифрової системи здійснюється за допомогою пристрою керування (ПК), який реалізує алгоритм роботи системи [3, 4]. Способи організації ПК та підходи до їх проєктування розглядаються в теорії цифрових автоматів [5–7].

Ключові характеристики ПК стосуються швидкодії, апаратурних витрат, габаритів, енергоспоживання, надійності, вартості тощо. Ці характеристики значною мірою визначають характеристики цифрової системи загалом [8, 9]. Проблематика теорії цифрових автоматів спрямована в тому числі і на оптимізації за зазначеними характеристиками з урахуванням умов проєктування. До таких умов належать, зокрема, алгоритм керування, елементний базис, тип та характеристики об'єкта керування, наявні засоби автоматизованого проєктування тощо. Найбільш варіативним можна вважати алгоритм керування, який імплементується схемою ПК. Відома велика кількість методів оптимізації пристроїв керування, ефективність яких залежить саме від структури і параметрів імплементованого алгоритму керування [10–12].

Алгоритм керування можна представити граф-схемою алгоритму (ГСА) [12, 13]. ГСА містить інформацію про стани автомата, порядок аналізу вхідних сигналів (логічних умов) та залежність вихідних сигналів (мікрооперацій) від вхідних сигналів та стану автомата. Такий рівень абстракції зазвичай є достатнім для опису поведінки ПК та синтезу його схеми в заданому елементному базисі. Довільна ГСА має як структурні атрибути – лінійна, вертикальна, зі зворотними зв'язками, розпаралелена тощо, так і набір числових параметрів, які визначаються її структурою та наповненням вершин. Під час автоматизованого проєктування може застосовуватись представлення ГСА у вигляді текстового файла спеціального формату, наприклад, у форматі KISS2 [14, 15].

Для проєктувальника ПК актуальною задачею є вибір методу оптимізації схеми пристрою. В загальному випадку вибір може здійснюватись не людиною, а спеціалізованою САПР. У низці випадків такий вибір може бути зроблений за результатами аналізу лише параметрів ГСА. Для цього треба знати, які параметри ГСА визначають ефективність певного методу схемотехнічної оптимізації, а які, навпаки, вказують про недоцільність використання цього методу. Такі знання є неочевидними і можуть бути отримані шляхом дослідження ефективності застосування тих чи інших методів схемної оптимізації до ГСА з різною структурою.

Для дослідження ефективності оптимізації схем пристроїв керування можуть бути використані тестові ГСА, що мають абстрактний вміст і відповідають певним вимогам. Водночас ефективність деяких методів може проявлятись лише у випадку ГСА високої складності, що містять сотні станів, вхідних та вихідних сигналів. Виникає науково-практична задача створення великої кількості тестових абстрактних ГСА високої складності із заданими параметрами. Розв'язання цієї задачі можливе шляхом псевдовипадкової генерації ГСА. Перш ніж розробляти відповідні алгоритми, необхідно проаналізувати відомі методи оптимізації та визначити набір параметрів ГСА, які необхідно враховувати в процесі генерації. Ця наукова робота присвячена задачі

формування набору параметрів (параметризації) граф-схем алгоритмів та оцінці їх взаємного впливу з огляду на псевдовипадкову генерацію ГСА згідно з цими параметрами.

Постановка завдання дослідження

Науковим завданням цієї роботи є систематизація і формування множини параметрів ГСА з метою подальшої генерації абстрактних ГСА у псевдовипадковий спосіб відповідно до заданих критеріїв.

Для розв'язання задачі насамперед необхідно визначитися з поняттям і загальними властивостями ГСА. Будемо вважати, що ГСА – це орієнтований зв'язний граф, який може містити чотири типи вершин: початкову, кінцеву, операторну та умовну [4, 7]. Початкова вершина відповідає початку алгоритму і не має входу. Кінцева вершина відповідає кінцю алгоритму і не має виходу. Операторна вершина містить набір одночасно виконуваних мікрооперацій, які надходять від ПК в об'єкт керування. Умовна вершина містить єдиний елемент із множини логічних умов (вхідних сигналів ПК) і має два виходи, що відповідають нульовому та одиничному значенням цієї умови.

Якщо вершини ГСА позначати символами v_i, то множина V вершин ГСА утворюється множиною операторних вершин V_Y, множиною умовних вершин V_X, початковою та кінцевою вершинами v_s, v_e:

$$V = V_Y \bigcup V_X \bigcup \{v_s,\ v_e\}. \quad (1)$$

Як приклад розглянемо ГСА G_1 (рис. 1). Без прив'язки до методу синтезу ПК цю ГСА можна охарактеризувати низкою базових параметрів [12, 13]:

Параметр 1. Загальна кількість вершин N_V=11. Цей параметр дорівнює кількості елементів множини V, яка, згідно з (1), для наведеного прикладу дорівнює $V = \{v_s, v_1,\ ..., v_9, v_e\}$. Окремо можна визначати кількість операторних вершин N_{V_Y} та кількість умовних вершин N_{V_X}. Для ГСА G_1: $N_{V_Y} = 5$ та $N_{V_X} = 4$.

Параметр 2. Кількість різних мікрооперацій – N_Y. Дорівнює потужності множини мікрооперацій Y. У нашому прикладі $Y = \{y_1,\ y_2,\ ...,\ y_7\}$, тому $N_Y = 7$.

Параметр 3. Кількість різних логічних умов – N_X. Дорівнює кількості елементів множини логічних умов X. У нашому прикладі $X = \{x_1,\ x_2,\ x_3\}$, тому $N_X = 3$. Обов'язково $N_X \le N_{V_X}$.

В операторних вершинах вказуються лише ті мікрооперації, які мають одиничні значення. Наприклад, у вершині v_8 мікрооперації y_2 та y_5 формуються рівними одиниці, усі інші, а саме y_1, y_3, y_4, y_6 та y_7, формуються рівними нулю. Для довільної ГСА визначимо параметри 4–7.

Параметр 4. Мінімальна кількість мікрооперацій в операторній вершині – $N_{Y_{min}}$.

Параметр 5. Максимальна кількість мікрооперацій в операторній вершині – $N_{Y_{max}}$.

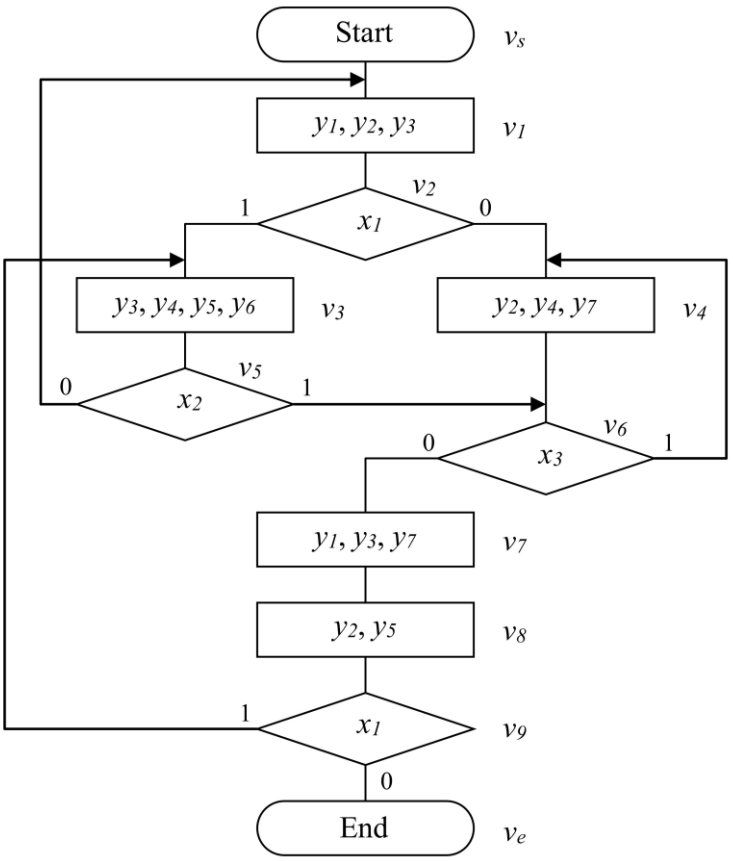

Рисунок 1. Граф-схема алгоритму G_1

Параметр 6. Середня кількість мікрооперацій в операторній вершині – $N_{Y_{mid}}$.

Параметр 7. Математичне очікування кількості мікрооперацій в операторній вершині – N_{Y_E} . N_{Y_E} є самостійним параметром і не пов'язаний безпосередньо із законом розподілу псевдовипадкових значень кількості мікрооперацій в операторних вершинах. Наприклад, якщо у згенерованої ГСА з параметрами $N_{Y_{min}} = 1$, $N_{Y_{max}} = 100$ N_{Y_E} дорівнює 10, то у більшості операторних вершин кількість мікрооперацій буде близька до N_{Y_E} .

Параметри 4–6 можуть залежати від використовуваного методу синтезу ПК, тому їх треба враховувати під час псевдовипадкового генерації ГСА. В загальному випадку можна вважати, що $N_{Y_{min}} > 0$, $N_{Y_{max}} \le N_Y$, $N_{Y_{min}} \le N_{Y_{mid}} \le N_{Y_{max}}$. Для ГСА G_1 $N_{Y_{min}} = 2$, $N_{Y_{max}} = 4$, $N_{Y_{mid}} = 3$.

Параметри 1–7 будемо розглядати як базові параметри ГСА, які не залежать від структури та методу синтезу ПК.

Параметризація ГСА відповідно до класу пристрою керування

Відомі наступні 3 класи ПК, що відрізняються способом імплементації алгоритму керування.

1. Мікропрограмний автомат (МПА) або автомат із жорсткою логікою [4, 7, 9, 13, 14, 16]. Імплементований алгоритм керування реалізується апаратно у вигляді електронної схеми за принципом кінцевого автомата.

2. Мікропрограмний пристрій керування (МПК) [12, 13, 17]. Алгоритм керування реалізується у вигляді набору мікрокоманд, що являють собою двійкові вектори і зберігаються у спеціальній керуючій пам'яті. Для доступу до керуючої пам'яті використовуються різні способи адресації мікрокоманд, що породжує різні структурні модифікації МПК та методи їх синтезу.

3. Композиційний МПК [12, 17, 18]. Структурно являє собою композицію мікропрограмного автомата і автомата з програмувальною логікою, в такий спосіб переваги обох цих класів ПК.

Розглянемо ці 3 класи ПК в контексті параметризації ГСА.

Мікропрограмний автомат

Під час синтезу ПК у вигляді мікропрограмного автомата (finite state machine, FSM) одним з етапів є розмітка ГСА відповідно до станів кінцевого автомата. Можливе позначення ГСА станами автомата Мура або станами автомата Мілі [4, 13, 14, 16]. У випадку автомата Мура відповідність його станів вершинам графу є такою:

– початкова і кінцева вершини позначаються однаковим станом a_0;
– кожна операторна вершина позначається окремим станом, починаючи з a_1.

У випадку автомата Мілі стани відповідають дугам ГСА так:

– вихід початкової і вхід кінцевої вершин позначаються однаковим станом b_0;
– вихід кожної операторної вершини позначається окремим станом, починаючи з b_1;
– якщо виходи декількох операторних вершин поєднуються, то вони позначаються єдиним станом – через це автомат Мілі може мати меншу кількість станів, ніж еквівалентний автомат Мура.

Для перевірки відповідності ГСА станами автомата Мілі треба переконатись, що під час переходу між будь-якими двома станами ми обов'язково проходимо через одну операторну вершину. Для ілюстрації на рис. 2 наведена ГСА G_1 з рис. 1 з розміткою станами автомата Мура (стани $a_0 - a_5$) та станами автомата Мілі (стани $b_0 - b_5$). Спосіб розмітки залежить від того, до якого типу кінцевих автоматів належить синтезований ПК.

Кількість станів автомата Мура або автомата Мілі не може бути обчислена аналітично на основі кількості вершин N_V, N_{V_Y} та N_{V_X}, або на основі інших розглянутих параметрів, оскільки потребує знання структури ГСА. Отже, кількість станів мікропрограмного автомата визначимо додатковим параметром.

Параметр 8. Кількість станів кінцевого автомата *M*.

Під час синтезу МПА його стани позначаються унікальними двійковими векторами (кодами станів). Розрядність кодів усіх станів у межах ГСА однакова і визначається так:

$$R = \log_2(M). \quad (2)$$

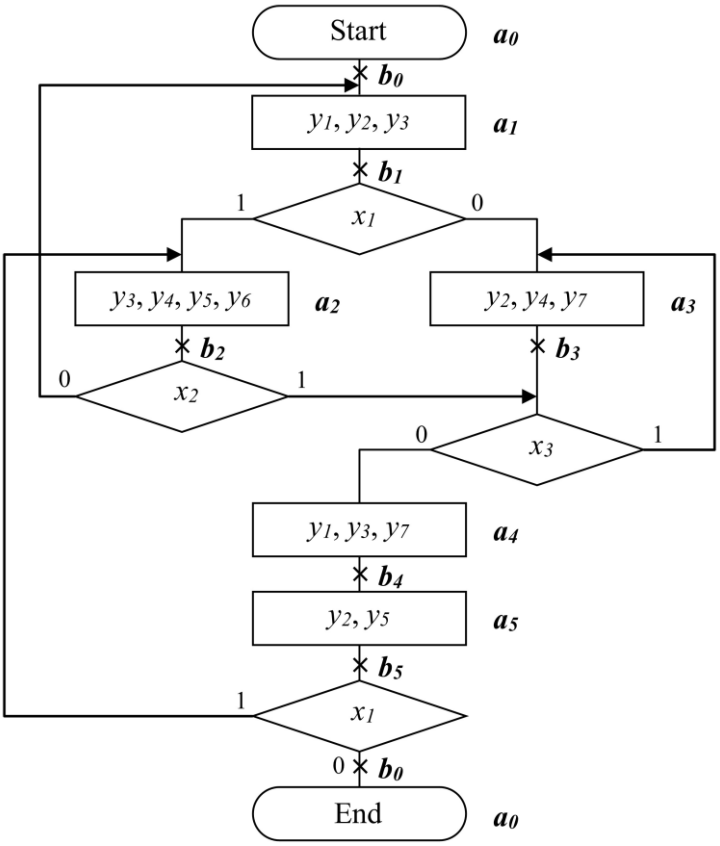

Рисунок 2. ГСА G_1 з розміткою станів автоматів Мілі і Мура

Якщо ГСА задана, значення R визначається значенням M. Для псевдовипадкової генерації ГСА значення R можна задавати і генерувати на його основі значення M. Наприклад, за $R=5$ кількість станів M генерованої ГСА може бути від 1 до 32, але з практичних міркувань M доцільно обмежити діапазоном від 17 до 32. Отже, будемо розглядати R як окремий параметр, хоча його використання за псевдовипадкової генерації ГСА не є обов'язковим і залежить від алгоритму генерації.

Параметр 9. Розрядність двійкових кодів станів автомата R.

Мікропрограмний пристрій керування

Цей клас ПК має регулярну структуру та відносно прості алгоритми синтезу. Метод синтезу залежить від використовуваного способу адресації мікрокоманд (примусова адресація, природна адресація, комбінована адресація та сторінкова адресація) [12, 17]. Замість станів МПА використовуються мікрокоманди, кожна з яких є кортежем булевих векторів певного формату.

В імплементованій ГСА можуть існувати переходи, які залежать від кількох логічних умов. На рис. 2 перехід зі стану a_2 в стан a_4 можливий за одночасного виконання умов $x_2=1,\ x_3=0$, тобто потребує аналізу двох логічних умов одночасно. Як наслідок, перехід зі стану a_2 можливий у більше ніж два стани, а саме у стани $a_1,\ a_3,\ a_4$. Такі переходи називаються багатоспрямованими (на відміну від однопрямованих та двоспрямованих) і в загальному випадку можуть залежати від будь-якої кількості логічних умов. Схема МПА

дає змогу реалізовувати будь-які багатоспрямовані переходи за один такт роботи пристрою. Однак формат мікро-команд МПК дає змогу в одній мікрокоманді вказувати код тільки однієї логічної умови. Залежно від значення логічної умови МПК виконує перехід по одній з двох можливих адрес, що задаються явно або неявно у відповідних полях мікрокоманди. Отже, МПК не здатен виконувати багатоспрямовані переходи за один такт своєї роботи.

Якщо задана ГСА містить послідовності умовних вершин, на першому етапі синтезу МПК необхідно виконати перетворення ГСА, позбавившись багатоспрямованих мікропрограмних переходів. Це робиться шляхом додавання порожньої операторної вершини в кожну дугу, що з'єднує дві умовні вершини. Порожня операторна вершина не містить жодної мікрооперації, однак відповідає окремій мікрокоманді в пам'яті МПК. Внаслідок цього перетворена ГСА буде містити більшу кількість вершин, ніж початкова ГСА, а кількість мікрокоманд МПК буде більшою, ніж кількість станів еквівалентного МПА Мура. Як наслідок, у випадку МПК виконання алгоритму керування за заданою ГСА може займати більшу кількість тактів, ніж у випадку МПА. Наприклад, якщо в ГСА є ділянки з десятьма послідовними умовними вершинами, МПА буде виконувати відповідні переходи за один такт, а МПК – за 10. Якщо такі ділянки в процесі роботи алгоритму повторюються циклічно, схема МПК може суттєво програвати у швидкодії схемі МПА. Відповідно до цих міркувань, введемо 3 додаткові параметри, які можуть бути використані для псевдовипадковій генерації ГСА з орієнтуванням на синтез ПК у вигляді МПК.

Параметр 10. Мінімальна кількість послідовно розташованих умовних вершин $N_{X_{min}}$.

Параметр 11. Максимальна кількість послідовно розташованих умовних вершин $N_{X_{max}}$.

Параметр 12. Середня кількість послідовно розташованих умовних вершин $N_{X_{mid}}$.

У загальному випадку для параметрів 10–12 справедливе таке відношення:

$$N_{X_{min}} \le N_{X_{mid}} \le N_{X_{max}} \le N_X . \quad (3)$$

Вираз (3) вказує, зокрема, на те, що у послідовно розташованих умовних вершинах не можуть зустрічатись однакові логічні умови. Якщо в ГСА міститься найдовший можливий ланцюжок умовних вершин довжиною N_X, усі логічні умови в цьому ланцюжку мають бути різними. На рис. 3 показано фрагмент абстрактної ГСА G_2, в якому у трьох послідовно розташованих умовних вершинах логічна умова x_1 зустрічається двічі. Це є помилкою, оскільки не дає змогу однозначно виконати перехід зі стану a_5 за умовою $x_1 = 0$.

Якщо необхідно, щоб у генерованій ГСА кількість послідовно розташованих умовних вершин групувалась навколо певного значення, можна за аналогією з параметром N_{Y_E}, ввести додатковий параметр 13.

Параметр 13. Математичне очікування кількості послідовно розташованих умовних вершин N_{X_E}.

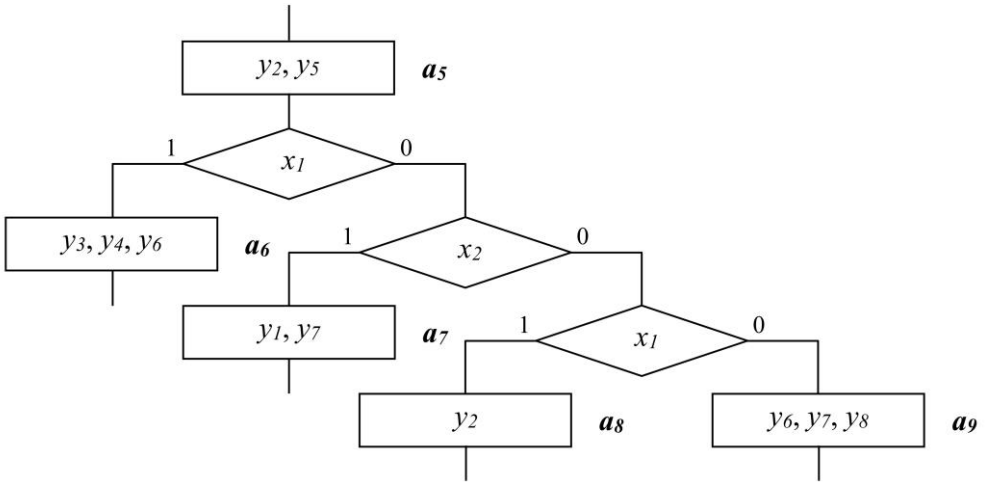

Рисунок 3. Фрагмент ГСА G_2, в якому у послідовно розташованих умовних вершинах двічі зустрічається логічна умова x_1

Композиційний мікропрограмний пристрій керування

Синтез композиційного МПК передбачає виявлення у заданій ГСА так званих операторних лінійних ланцюгів (ОЛЛ, operational linear chain) [18]. Із поняттям ОЛЛ пов'язані наступні чотири визначення.

1. Операторним лінійним ланцюгом у ГСА називається така кінцева послідовність операторних вершин, що для будь-якої пари вершин v_i та v_{i+1} існує дуга, що виходить з v_i і входить у v_{i+1}.

2. Входом ОЛЛ називається операторна вершина цієї ОЛЛ, вхід якої зв'язаний з початковою або умовною вершиною, або з вершинами, що входять в інші ОЛЛ.

3. Вхід ОЛЛ називається головним входом, якщо відсутній зв'язок цього входу з виходами операторних вершин.

4. Виходом ОЛЛ називається вершина, вихід якої зв'язаний зі входами кінцевої або умовної вершини, або зі входом операторної вершини, що входить в іншу ОЛЛ.

Також існує поняття простого операторного лінійного ланцюга. Простим ОЛЛ називається послідовність операторних вершин, між якими немає умовних вершин, а вхід кожної вершини (окрім початкової) зв'язаний тільки з виходом попередньої вершини цього ОЛЛ. Входом простого ОЛЛ вважається операторна вершина, вхід якої зв'язаний з виходом початкової або умовної вершини, або з виходами декількох вершин (операторних чи умовних). Виходом простого ОЛЛ вважаться операторна вершина, вихід якої зв'язаний з кінцевою або умовною вершиною, або з операторною вершиною, що є входом іншої ОЛЛ.

На рис. 4 зображена абстрактна ГСА G_3, в якій операторні лінійні ланцюги окреслені пунктиром. ГСА містить чотири ОЛЛ $O_1 - O_4$. ОЛЛ O_2 та O_4 є простими, оскільки містять одну вхідну вершину. ОЛЛ O_1 містить дві вхідні вершини v_1 та v_2, ОЛЛ O_3 також містить дві вхідні вершини v_8 та v_{10}.

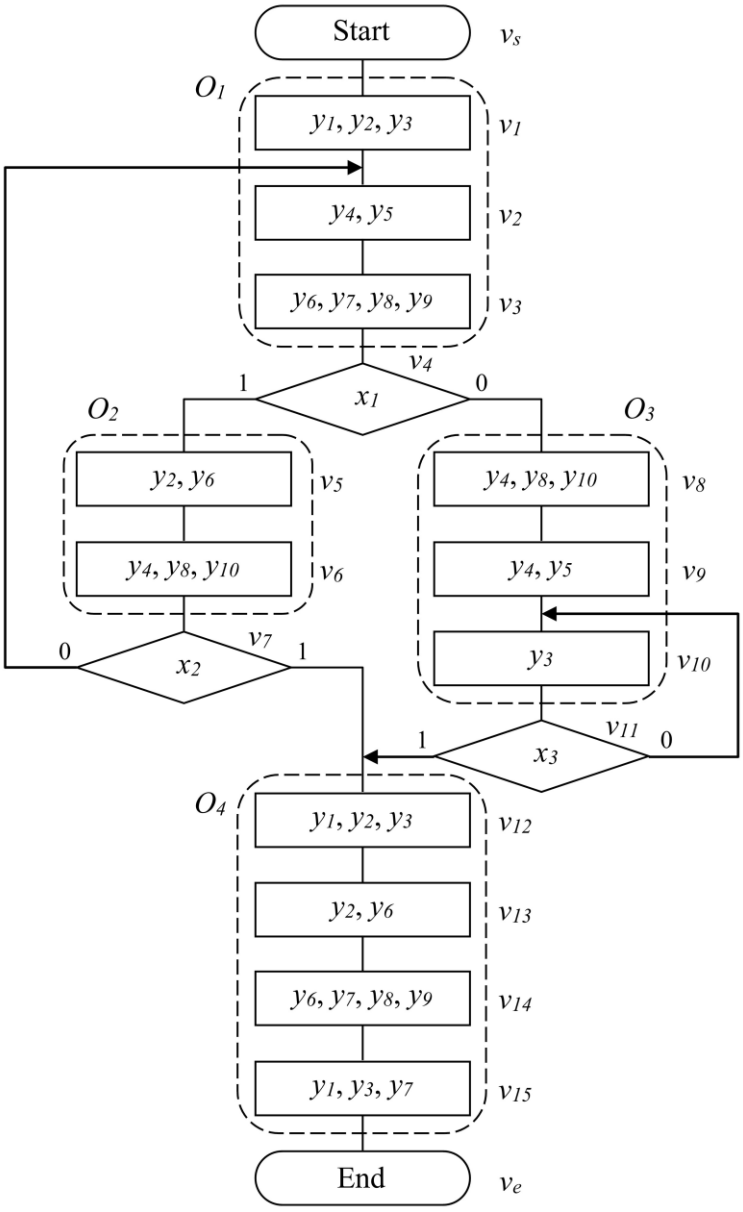

Рисунок 4. ГСА G_3 з виділеними операторними лінійними ланцюгами

ГСА G_1 з рис. 1 також містить 4 ОЛЛ, хоча й меншої довжини. В алгоритмі генерації псевдовипадкової ГСА, спрямованої на синтез КМПК, пропонується враховувати такі параметри.

Параметр 14. Кількість ОЛЛ в ГСА N_{OLC}. В загальному випадку $N_{OLC} \le N_{V_Y}$.

Параметр 15. Мінімальна довжина ОЛЛ $N_{OLC_{min}}$.

Параметр 16. Максимальна довжина ОЛЛ $N_{OLC_{max}}$.

Параметр 17. Середня довжина ОЛЛ $N_{OLC_{mid}}$.

Параметр 18. Математичне очікування довжини ОЛЛ N_{OLC_E}.

Параметр 19. Мінімальна кількість входів ОЛЛ $N_{ENT_{min}}$.

Параметр 20. Максимальна кількість входів ОЛЛ $N_{ENT_{max}}$.

У загальному випадку $1 \le N_{ENT_{min}} \le N_{ENT_{max}}$, $N_{OLC_{min}} \le N_{ENT_{max}} \le N_{OLC_{max}}$. За умови $N_{ENT_{max}} = 1$ усі ОЛЛ в ГСА стають простими.

Параметризація ГСА для різних методів оптимізації

Більшість методів оптимізації пристроїв керування спрямована на зменшення їх апаратурних витрат, що дає змогу знизити вартість і фізичні розміри схеми, підвищити її енергоефективність, надійність та швидкодію. Дослідження ефективності того чи іншого методу оптимізації проводяться зазвичай в умовах використання елементного базису, який є актуальним на момент розробки методу. У разі зміни елементного базису ефективність методів оптимізації схеми ПК треба перевіряти повторно. Також майже не дослідженою залишаються можливість одночасного застосування різних методів оптимізації та їх взаємний вплив. Такі дослідження можуть бути проведені з використанням множини ГСА, згенерованих у псевдовипадковий спосіб відповідно до заданих параметрів.

Розглянемо методи оптимізації, які спрямованими на мінімізацію апаратурних витрат у схемі ПК, зокрема: кодування наборів мікрооперацій, заміна логічних умов, кодування автоматних переходів, вертикалізація ГСА, формування класів псевдоеквівалентних станів, використання лічильника кодів станів та операційне перетворення кодів станів.

Кодування наборів мікрооперацій

Набором мікрооперацій називається множина мікрооперацій, що записана в одній операторній вершині ГСА [12, 16]. За своєю суттю набір мікрооперацій поступає в об’єкт управління і впливає одночасно на різні його елементи з метою виконання складної операції. Якщо об’єктом управління ПК виступає операційний автомат або арифметико-логічний пристрій, кожен набір мікрооперацій може ініціювати одну операцію над даними, як-от додавання, віднімання, кон’юнкція тощо. Протягом виконання свого алгоритму пристрій керування може неодноразово ініціювати виконання однієї і тієї ж дії об’єктом керування. На рівні ГСА це проявляється в тому, що декілька операторних вершин містять один і той самий набір мікрооперацій.

На рис. 4 ГСА G_3 містить 7 різних наборів мікрооперацій: $Y_1 = \{y_1, y_2, y_3\}$, $Y_2 = \{y_4, y_5\}$, $Y_3 = \{y_6, y_6, y_8, y_9\}$, $Y_4 = \{y_2, y_6\}$, $Y_5 = \{y_4, y_8, y_{10}\}$, $Y_6 = \{y_3\}$, $Y_7 = \{y_1, y_3, y_7, y_4\}$. Для кодування семи наборів достатньо трьох двійкових розрядів. Внаслідок цього схема ПК формує 3 розряди коду набору мікрооперацій замість 10 окремих мікрооперацій $y_1 - y_{10}$, після чого проводить дешифрування кодів наборів та формує окремі мікрооперації. Це за певних умов дає змогу зменшити складність схеми, що реалізує функцію виходів ПК.

Псевдовипадкова генерація наборів мікрооперацій потребує таких параметрів:

Параметр 21. Кількість наборів мікрооперацій у ГСА N_{SET}^{Y}. Цей параметр не може перебільшувати 2^{N_Y} (потужність булеану для множини мікрооперацій).

Параметр 22. Мінімальна кількість мікрооперацій в наборі $N^Y_{SET_{min}}$.

Параметр 23. Максимальна кількість мікрооперацій в наборі $N^Y_{SET_{max}}$.

Параметр 24. Середня кількість мікрооперацій в наборі $N^Y_{SET_{mid}}$.

Параметр 25. Математичне очікування кількості мікрооперацій в наборі $N^Y_{SET_E}$.

У загальному випадку $0 \le N^Y_{SET_{min}} \le N^Y_{SET_{max}}$, $N^Y_{SET_{min}} \le N^Y_{SET_{max}} \le N_Y$.

Заміна логічних умов

Заміна логічних умов застосовується в тих випадках, коли в ГСА аналізується велика кількість різних логічних умов, однак одночасно (в одному переході) аналізується лише декілька із них [12, 16–18]. Наприклад, у будівлі є 100 кімнат, в кожній з яких встановлені 3 датчики: датчик руху, датчик вібрації та датчик тепла. Пристрій керування сигналізацією циклічно перевіряє усі датчики в кожній кімнаті по черзі. Цикл перевірки кімнат повторюється нескінченну кількість разів. У разі спрацювання якогось датчика ПК реагує у певний спосіб. ГСА, що реалізує подібний алгоритм, буде мати 300 вхідних сигналів $x_1 - x_{300}$, однак одночасно будуть перевірятись лише 3, що відповідають датчикам окремої кімнати. Метод заміни вхідних змінних дає змогу використати в ГСА замість 300 логічних умов 3 псевдологічні умові p_1, p_2, p_3, які можуть перевірятись одночасно в одному мікропрограмному переході. Під час цього до структури ПК вводиться додаткова схема, яка перетворює 3 логічні умови, потрібні для аналізу на даному кроці, у сигнали p_1, p_2, p_3. Внаслідок цього замість 300 сигналів від логічних умов у схему ПК подаються 3 сигнали, що знижує складність схеми ПК. Кількість сигналів p_i визначається максимальною кількістю логічних умов, що аналізуються одночасно в межах ГСА. У ГСА G_1 на рис. 1 одночасно аналізуються максимум 2 логічні умови, а в ГСА G_3 на рис. 4 – лише одна логічна умова.

У ГСА кількість одночасно аналізованих логічних умов дорівнює кількості послідовно розташованих умовних вершин. Однак ми вже визначали подібні параметри під час розгляду мікропрограмних пристроїв керування. Отже, під час генерації ГСА зі спрямуванням на метод заміни вхідних змінних, можна використовувати раніше визначені параметри 10–13 та враховувати співвідношення (3). Потреби у введенні окремих параметрів ГСА для цього метода немає.

Кодування переходів автомата

Під час синтезу мікропрограмного автомата будується пряма структурна таблиця, кожен рядок якої відповідає окремому переходу автомата з одного стану в інший. Рядки прямої структурної таблиці кодуються унікальними двійковими кодами, на основі яких спеціальна схема формує відповідний код стану переходу та набір мікрооперацій. Такий підхід дає змогу спростити схему адресації і за певних умов зменшити апаратурні витрати на реалізацію схеми автомата [12, 16].

Переходи можуть бути безумовними (прямими) і умовними. Безумовний перехід не передбачає перевірки логічних умов і в ГСА виглядає як дуга між двома операторними

вершинами або між початковою і операторною вершинами чи між операторною і кінцевою вершинами. Умовний перехід – це послідовність із мінімум трьох послідовно зв'язаних вершин, у якій:

- першою вершиною є початкова або операторна вершина;
- останньою вершиною є операторна або кінцева вершина;
- між першою і останньою вершинами розташовані одна або декілька умовних вершин.

Очевидним параметром ГСА, що враховується методом кодування переходів, є загальна кількість переходів з одного стану в інший. Також можна окремо розглядати кількість або частку умовних і безумовних переходів, що опишемо такими трьома параметрами:

Параметр 26. Кількість переходів N_T.

Параметр 27. Кількість безумовних переходів N_{T_D}.

Параметр 28. Кількість умовних переходів N_{T_C}.

Вертикалізація ГСА

Суть вертикалізації полягає у приведенні ГСА до такого вигляду, що в кожній операторній вершині міститься не більше однієї мікрооперації [12, 16]. Така ГСА називається вертикальною і може бути отримана з початкової ГСА шляхом розщеплення операторних вершин із декількома мікрооперaціями на відповідну кількість послідовно розташованих операторних вершин з однією мікрооперацією в кожній. Далі усі мікрооперації у ГСА кодуються унікальними двійковими кодами. Ці коди формуються спеціальним блоком у схемі ПК, після чого здійснюється їх декодування і отримання сигналів мікрооперацій, що надходять в об'єкт керування.

Для формування вертикальної ГСА можна скористатись раніше введеними параметрами 4–7, взявши їх усі рівними одиниці. З іншого боку, можна не генерувати вже вертикалізовану ГСА, залишивши процес вертикалізації відповідному алгоритму синтезу автомата. Отже, додаткових параметрів ГСА у випадку зазначеного методу вводити не потрібно.

Формування класів псевдоеквівалентних станів

Формування класів псевдоеквівалентних станів застосовується тільки для МПА Мура і дає змогу замінити стани автомата класами, кількість яких менша за кількість станів автомата Мура і дорівнює кількості станів автомата Мілі [12, 13, 16]. Зменшення апаратурних витрат у схемі автомата можливе тоді, коли для кодування класів псевдоеквівалентних станів потрібна менша кількість розрядів, ніж для кодування станів автомата Мура.

В автоматі Мура псевдоеквівалентними вважаються стани, що відповідають операторним вершинам, виходи яких поєднані зі входом однієї вершини (операторної, умовної або кінцевої). На рис. 5 наведена ГСА G_4, в якій можуть бути виділені чотири класи псевдоеквівалентних станів: $B_1 = \{a_0\}$, $B_2 = \{a_1, a_2\}$, $B_3 = \{a_3, a_4, a_5\}$, $B_4 = \{a_6\}$. Для кодування чотирьох класів $B_1 - B_4$ достатньо двох двійкових розрядів, тоді як для

кодування семи станів $a_0 - a_6$ потрібно 3 розряди. Менша кількість розрядів у загальному випадку сприяє спрощенню схеми адресації МПА.

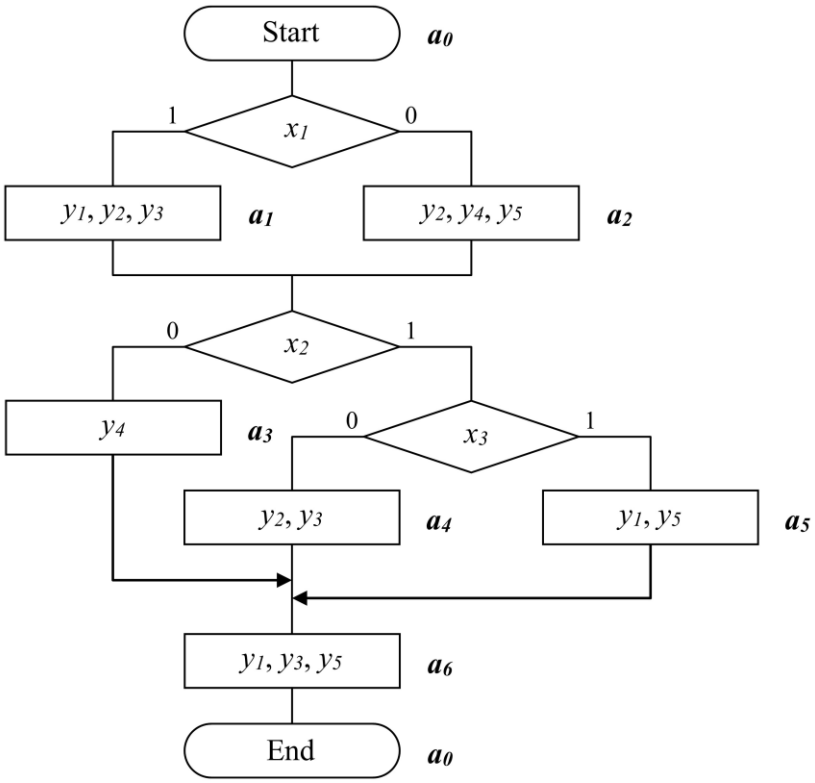

Рисунок 5. Граф-схема алгоритму G_4

Введемо параметри, які характеризують ГСА з погляду об'єднання виходів операторних вершин.

Параметр 29. Мінімальна кількість операторних вершин з об'єднаними виходами $N_{B_{min}}$.

Параметр 30. Максимальна кількість операторних вершин з об'єднаними виходами $N_{B_{max}}$.

Параметр 31. Середня кількість операторних вершин з об'єднаними виходами $N_{B_{mid}}$.

Параметр 32. Математичне очікування кількості операторних вершин з об'єднаними виходами N_{B_E}.

Використання лічильника кодів станів

Структура композиційного мікропрограмного пристрою керування містить у своєму складі лічильник мікрокоманд [12, 16]. Він реалізує природну адресацію мікрокоманд у межах операторних лінійних ланцюгів. Подібний підхід може бути застосований до мікропрограмних автоматів і породжує клас автоматів на лічильнику, в яких лічильник використовується замість звичайного регістра пам'яті [16, 18].

За аналогією з операторними лінійними ланцюгами в автоматах на лічильнику визначаються лінійні послідовності станів (ЛПС). ЛПС відрізняється від ОЛЛ тим, що в межах ЛПС допустиме існування як безумовних переходів, так і умовних переходів за одиничним значенням логічної умови. Такі переходи в автоматах на лічильнику виконуються за допомогою інкремента. На рис. 4 перехід із вершини v_3 у вершину v_4 здійснюється за умови $x_1 = 1$. Так само виконується перехід із вершини v_6 у вершину v_{12}. Якщо ГСА G_3 позначити станами автомата Мура, то стани, відповідні до вершин v_1, v_2, v_3, v_5, v_6, v_{12}, v_{13}, v_{14}, v_{15}, утворюють ЛПС і для коректної роботи автомата мають бути закодовані послідовними кодами. Друга ЛПС ГСА G_3 буде містити стани автомата Мура, еквівалентні вершинам v_8, v_9, v_{10}. Ці стани також мають бути закодовані послідовними кодами, відмінними від кодів станів першої ЛПС. Отже, в ГСА G_3 можуть бути виділені 4 операторні лінійні ланцюги та 2 лінійні послідовності станів. Ці структурні складники не пов'язані між собою і використовуються в різних методах синтезу.

Постає питання: чи є потреба у введенні додаткових параметрів, що характеризують ГСА в контексті ЛПС, чи запропонованих для ОЛЛ параметрів 14–20 достатньо? ОЛЛ і ЛПС – різні поняття, і їх генерація має відбуватися за різними алгоритмами. ОЛЛ і ЛПС використовуються в різних класах пристроїв керування. Генерація ГСА, в якій одночасно будуть *X* ОЛЛ та *Y* ЛПС, не завжди є можливою і не має практичної цінності. Якщо потрібно порівняти ефективності КМПК і автомата на лічильнику, достатньо згенерувати ГСА із заданою кількістю ОЛЛ або із заданою кількістю ЛПС, після чого синтезувати по цій ГСА обидва типи ПК і порівняти характеристики отриманих схем. Отже, для генерування ГСА для автоматів на лічильнику можна використовувати параметри 14–20, розуміючи під ОЛЛ саме лінійні послідовності станів. Впровадження додаткових параметрів ГСА в цьому випадку є зайвим.

Операційне перетворення кодів станів

Операційне перетворення кодів станів використовується в мікропрограмних автоматах. Переходи між станами автомата здійснюються шляхом виконання арифметико-логічних операцій над кодом поточного стану [19, 20]. Сьогодні не визначено параметри ГСА, які сприяють ефективності автоматів з операційним перетворенням кодів станів. Результатом синтезу таких пристроїв є зіставлення кожному автоматному переходу однієї арифметико-логічної операції із заданої множини операцій відповідно до певних вимог. Можна стверджувати, що чим більше переходів містить ГСА, тим складнішим є синтез такого МПА. Отже, саме кількість переходів ГСА постає визначальним критерієм. Це дає змогу спиратись під час псевдовипадкової генерації ГСА на раніше введені параметри 26–28 і не вводити додаткові параметри.

Обговорення отриманих результатів

Запропоновані 32 параметри граф-схем алгоритмів дають змогу описати ГСА в контексті різних класів пристроїв керування та методів оптимізації їх схем. Ці параметри можуть бути використані як початкові дані для програм псевдовипадкового генерування

граф-схем алгоритмів. Перелік параметрів не є остаточним і може доповнюватися з урахуванням інших структур пристроїв керування та методів їх синтезу.

Однак потрібно враховувати, що деякі з розглянутих параметрів можуть бути взаємовиключними. Наприклад, неможливо сформувати $N_{SET}^{Y} = 100$ наборів мікрооперацій, якщо загальна кількість різних мікрооперацій в ГСА $N_Y = 5$. Якщо користувач зазначив саме такі значення параметрів, програма, що генерує ГСА, повинна повідомити про помилку або запропонувати найбільш близькі значення параметрів, що не викликають конфлікту.

Генерація псевдовипадкових ГСА за заданими параметрами не є тривіальною задачею. Окрім відповідності параметрам, згенерована ГСА повинна мати такі загальні ознаки коректності, наприклад:

- не містити вершин, із яких неможливо досягти кінцевої вершини;
- не містити переходів у початкову вершину (замість них треба використовувати переходи у кінцеву вершину);
- не містити вершин, вхід або вихід яких не зв’язані з іншими вершинами або зв’язані з неіснуючими вершинами;
- не містити ланцюгів умовних вершин, у яких хоча б одна логічна умова зустрічається більше одного разу;
- містити рівно одну початкову та кінцеву вершини та хоча б одну операторну вершину.

Подібні перевірки рекомендується робити відразу після генерації ГСА. У разі виявлення помилок бажано мати можливість їх автоматичного виправлення. Також корисною функцією може бути візуалізація згенерованої ГСА з висвітленням тих чи інших особливостей генерації (наприклад, із виділенням операторних лінійних ланцюгів). Візуалізація ГСА може виконуватись окремим програмним модулем, початковими даними для якого виступатиме текстовий файл з описом згенерованої ГСА.

Дослідження методів синтезу цифрових пристроїв керування зазвичай потребують великої кількості тестових ГСА. Автоматизація методів синтезу дає змогу синтезувати ПК відповідно до ГСА, що містять сотні і тисячі станів або мікрокоманд. Псевдовипадкова генерація ГСА за заданими параметрами дасть можливість швидкого створення великої кількості різних ГСА зі схожими характеристиками, що зекономить час та підвищить якість і достовірність отриманих результатів.

Висновки

Сформовано множини уніфікованих параметрів граф-схем алгоритмів із метою використання їх у процесі псевдовипадкової генерації ГСА. Вибрані параметри враховують як базові структурні властивості ГСА, так і особливості різних класів ПК та методів оптимізації схем пристроїв.

Практична цінність роботи полягає у закладанні підґрунтя для створення алгоритмів і програмних засобів псевдовипадкової генерації ГСА, що дають змогу дослідити ефективність відомих та нових методів синтезу й оптимізації схем пристроїв керування.

Подальший напрям досліджень автори вбачають у розробці і дослідженні алгоритмів псевдовипадкової генерації ГСА відповідно за запропонованими в цій роботі множині параметрів.

Література

1. Bailliul, J., Samad, T. (2015). *Encyclopedia of Systems and Control*. Springer: London, UK, 1554 p. https://doi.org/10.1007/978-1-4471-5058-9
2. Suh, S. C., Tanik, U. J., & Carbone, J. N. (2013). *Applied Cyber-Physical Systems*. New York, USA: Springer, 253 p. https://doi.org/10.1007/978-1-4614-7336-7
3. Kam, T., Villa, T., Brayton, R., & Sangiovanni-Vincentelli, A. (1997). *A Synthesis of Finite State Machines: Functional Optimization*. Boston, MA, USA: Springer, 282 p. https://doi.org/10.1007/978-1-4757-2622-0
4. Baranov, S. (1994). *Logic Synthesis for Control Automata*. Dordrecht: Kluwer Academic Publishers, 312 p.
5. Hartmanis, J., Stearns, R. E. (1966). *Algebraic Structure Theory of Sequential Machines*. New Jersey: Prentice-Hall, 211 p.
6. Czerwinski, R., Kania, D. (2013). *Finite State Machine Logic Synthesis for Complex Programmable Logic Devices*. Berlin: Springer, 172 p. https://doi.org/10.1007/978-3-642-36166-1
7. Sklyarova, I., Sklyarov, V., & Sudnitson, A. (2012). *Design of FPGA-based Circuits using Hierarchical Finite State Machines*. Tallin: TUT Press, 240 p.
8. Nelson, V., Nagle, H., Irwin, J., & Carroll, B. (1995). *Digital logic circuit analysis and design*. New Jersey: Prentice Hall, 842 p.
9. DeMicheli, G. (1994). *Synthesis and Optimization of Digital Circuits*. NY: McGraw-Hill, 576 p.
10. Baranov, S. (2008). *Logic and System Design of Digital Systems*. TUTPress: Tallinn, Estonia, 267 p.
11. Scholl, C. (2001). *Functional Decomposition with Application to FPGA Synthesis*. Boston, MA, USA: Kluwer Academic, 264 p. https://doi.org/10.1007/978-1-4757-3393-8
12. Barkalov, A. A., Titarenko, L. A., Babakov, R. M., & Baiev, A. V. (2016). *Logic synthesis for FPGA-based finite state machines: monograph*. Vinnytsia: DonNU Vasyl' Stus, 195 p.
13. Baranov, S. (2018). *Finite State Machines and Algorithmic State Machines*. Seattle, WA, USA: Amazon, 185 p.
14. Yang, S. (1991). *Logic synthesis and optimization benchmarks User Guide Version 3.0.* Tech. rep. Microelectronics Center of North Carolina, P. O. Box 12889, Research Triangle Park, NC 27709.
15. Minns, P., Elliot, I. (2008). *FSM-Based Digital Design Using Verilog HDL*. New Jersey: J. Wiley & Sons, 408 p.
16. Barkalov, A., Titerenko, L. (2009). *Logic synthesis for FSM-based control units*. Berlin: Springer, 233 p.

17. Barkalov, A., Wegrzyn, M. (2006). *Design of control units with programmable logic*. Zielona Gora: University of Zielona Gyra Press, 150 p.
18. Barkalov, A., Titerenko, L. (2008). *Logic synthesis for compositional microprogram control units*. Berlin: Springer, 272 p.
19. Babakov, R. M. (2019). *Development and Hardware Optimization of Control Units for IOT Devices in Industry Systems*. Internet of Things for Industry and Human Applications. Vol. 1. Assessment and Implementation / V. S. Kharchenko (ed.). Ministry of Education and Science of Ukraine, National Aerospace University KhAI, p. 834–865.
20. Barkalov, A. A., Titarenko, L. A., & Babakov, R. M. (2023). *Synthesis of VHDL-Model of a Finite State Machine with Datapath of Transitions*. Radio Electronics, Computer Science, Control. Volume 4, p. 135–147. https://doi.org/10.15588/1607-3274-2023-4-13

Рукопис отримано – 04/03/2025 р.; прийнято до публікації – 25/03/2025 р.

Parametrization of graph-schemes of algorithms for digital control units

Roman Babakov, Alexander Barkalov

Abstract

The paper considers the scientific and practical problem of determining the set of parameters of graph-schemes of algorithms in the context of further pseudo-random generation of graph-schemes in order to study the effectiveness of methods for synthesizing and optimization of digital control units. Structural components are considered and general parameters of graph-schemes of algorithms are determined, which are traditionally used to describe algorithms for digital control units. The main classes of digital control units are analyzed, such as microprogram finite state machine (automated state machine with hardware logic implementation), microprogram control unit (automated state machine with programmable logic) and compositional microprogram control unit. For these classes, the main methods of optimizing hardware expenses are considered, including encoding sets of microoperations, replacing input variables, operational transformation of state codes, etc. For each of the considered classes of control units and optimization methods, sets of graph-schemes of algorithms parameters are proposed, which affect the effectiveness of applying the corresponding structures and methods and characterize both the transition function and the function of the outputs of control unit. For individual parameters, the permissible range of changes, correlation or mutual exclusivity with other parameters of graph-schemes are determined. Illustrative examples of determining individual parameters for a given graph-scheme are given. Recommendations are given on the use of the proposed parameters for pseudo-random generation of graph-schemes of algorithms. General requirements for correct pseudo-random generation of graph-schemes of algorithms are determined. Such requirements are: the possibility of reaching the final node from any other node; the absence of nodes whose input is not connected with the output of another node; the absence of repetition of logical conditions in consecutive conditional nodes; the presence of at least one operational node, etc.

Keywords: graph-scheme of algorithm; digital control units; circuit optimization; parameters; pseudo-random generation.