Зовнішніми входами вважають входи елементів верхньої границі, а зовнішніми виходами вважатимемо виходи елементів правої границі.

Запропонована схема (рис.3), виконує такі функції. У вертикальному напрямку розповсюджується двійковий сигнал y, який не змінюється елементами схеми. В горизонтальному напрямку розповсюджується зліва направо сигнал s. Якщо на вхід лівого елемента стрічки подати сигнал $s_0 = 1$, то значення міжелементних сигналів залежатимуть від співвідношення x і y:

$$s' = s(xy \vee \overline{xy}) .$$

Доти, поки $x \equiv y$, зберігається $s = 1$. В елементі, в якому вперше $x_i \neq y_i$, на виході з'явиться сигнал $s = 0$, який вже не може змінитися до правої границі схеми.

Очевидно, що якщо на входи y_0^i подати двійкове слово Y, а на всі входи S_0 лівої границі – сигнал $s = 1$, то на правих виходах схеми сигнал $s = 1$ буде лише тоді, якщо $X = Y$. Тобто $x_i = y_i$ для всіх розрядів слова. Отже, така схема виконує функції асоціативної пам'яті. Вона може бути реалізована на одному чіпі, або за допомогою ООС[2].

1. Деркач Б.Т. Застосування асоціативної обробки інформації для синтезу нових обчислювальних структур // Відбір і обробка інформації. 1997. Вип.11. С.111-114

2. Деркач Б.Т. Реалізація асоціативних обчислювальних структур на ООС //Відбір і обробка інформації. 1998, Вип.12. С.139-144

3. Reid M.M., Millar R.J., Black N.D. Second - Generation Image Coding. An Overview //ACM Computing Surveys. 1997, 29, No.1. P.3-30

4. Shooman W. Orthogonal Processing. In Parallel Processors System Technologies and Applications. Spartan Books, 1979. P.297-308.

УДК 621.382

АЛГОРИТМІЧНА РЕАЛІЗАЦІЯ КОНСТРУКТИВНОГО РОЗБИТТЯ СХЕМ

© Р. Базилевич, І. Подольський
Національний університет "Львівська політехніка"

Розглянуто нові алгоритми конструктивного розбиття схем на основі методу оптимального згортання схеми. Деталізовано їх застосування для розбиття схем на дві частини з адаптацією для швидкодіючої програмної реалізації.

New algorithms of constructive circuits partitioning are presented on the basic of the Optimal Circuit Reduction method. They realize 2-way circuit partitioning with adaptation for high-speed software implementation.

Вступ

Сучасні технології вимагають застосування схем великої розмірності, прототипи яких реалізуються на програмованих логічних матрицях ПЛМ (*FPGA*). Задача оптимального призначення елементів в окремі матриці є NP складною. Для отримання якісної декомпозиції схеми, на задану кількість частин, застосовуються різні алгоритми розбиття схеми на частини. Розглянемо один з таких алгоритмів, що базується на методі оптимального згортання схеми (ОЗС) [1].

Формулювання задачі

Необхідно отримати розбиття $\tilde{P}$ схеми $S = (P, E)$, де P – множина елементів, E – множина зв'язків між елементами, на дві рівні частини, щоб мінімізувати кількість зв'язків між ними:

$$P \to \tilde{P} = \{P_1, P_2\}, |P_1| = |P_2|, n^{ex}(\tilde{P}^*) \to \min_i n^{ex}(\tilde{P}_i).$$

Розбиття схеми на частини складається з двох етапів. Початковий етап дає наближені результати, які на другому етапі оптимізуються. Етап початкового розбиття називатимемо надалі конструктивним розбиттям. Суть алгоритму полягає в тому, що спочатку на основі паралельно-послідовного згортання схеми виділяються природні підсхеми, які є вершинами дерева оптимального згортання T^R. У результаті згортання отримуємо ліс дерев $F^T = \{T^R_1, ..., T^R_n\}$. На основі аналізу лісу дерев розділяємо схему на частини. Надалі розглядатимемо декомпозицію схеми на дві частини (бінарне розбиття).

Розділ схеми відбувається так, щоб заповнити дві частини однаковою або приблизно однаковою кількістю елементів. Призначаємо отримані дерева частинам схеми, а дерево, що матиме надлишкову кількість елементів, розділяється на дві частини із заданою кількістю елементів у кожній з них. Для виконання цієї задачі необхідно знайти на дереві кластер з кількістю елементів, що приблизно дорівнює тій, що її необхідно відділити. Після цього розглядатиметься задача оптимізації розміру відділеної частини та приведення його до заданого, шляхом перенесення за певними критеріями кластерів між підсхемами .

Алгоритми початкового розбиття

Введемо поняття головної частини розбиття та доповнювальної для спрощення програмної реалізації алгоритму. Приймаємо частину P_1 головною.

Для виділення сильнозв'язаних частин схеми застосовуємо до заданої схеми S ієрархічну кластеризацію [2], в результаті чого отримуємо ліс дерев F^T, який в більшості випадків складатиметься з одного дерева. Якщо схема містить кілька незв'язаних між собою підсхем, то в результаті ієрархічної кластеризації отримуємо кілька дерев. Для задачі призначення елементів схеми у дві окремі частини у випадку з кількома деревами пропонується такий алгоритм наповнення частин:

1. Представляємо кожну групу елементів, що утворила окреме дерево, його кореневою вершиною, тобто кластером з потужністю, що відповідає кількості елементів підсхеми.
2. Будуємо список з інформацією про кореневу вершину кожного дерева.
3. Впорядковуємо список кореневих вершин дерев за спаданням їх потужностей. Коренева вершина найбільшого дерева буде першою у списку, що дозволяє прийняти рішення щодо вибору гілки алгоритму для продовження роботи.
4. Вибираємо першу вершину зі списку та визначаємо її розмірність з метою призначення множини елементів відповідного дерева (підсхеми) частині P_1 розбиття або для проведення декомпозиції цієї підсхеми на основі дерева оптимального згортання.

   а) якщо потужність дерева, відповідного до вибраної вершини, є більшою за ємність головної частини, яку необхідно наповнити, то застосовується процедура віднімання надлишку елементів $N = |T^{\kappa}_1| - |P_1|$, а елементи усіх інших дерев призначаються у доповнювальну частину розбиття P_2. Головна частина розбиття буде наповнена елементами вибраного найбільшого дерева, а надлишок переноситься у доповнювальну частину згідно з процедурою віднімання надлишку. У частковому випадку, коли ліс дерев міститиме лише одне дерево, задача віднімання надлишку зведеться до декомпозиції дерева на дві частини. Переходимо на крок 5.

   б) якщо потужність виділеного дерева дорівнює за розміром одній з частин, тоді задача призначення розв'язана. Елементи цього дерева наповнюють відповідну частину, а елементи дерев, що залишились, призначаються в іншу частину. Переходимо на крок 5.

   в) якщо потужність дерева менша за кількість елементів, якими треба наповнити головну частину P_1, то елементи дерева призначаються в цю частину. Цей крок актуальний для випадків, коли отримуємо в результаті кластеризації ліс з великою кількістю дерев. Відповідно, кількість елементів, якою необхідно наповнити першу частину, зменшується на величину потужності призначеного дерева. Якщо частину P_1 ще необхідно доповнити, то обираємо наступну вершину зі списку і повторюємо цей крок до наповнення частини P_1. Якщо при призначенні чергового дерева виявиться, що заповнення відбудеться з надлишком і виникне задача надбирання надлишку, то застосовується процедура віднімання надлишку. Для зменшення надлишку і відповідно складності обчислень можна оминути вибрану вершину і перейти до розгляду наступної, що відповідає дереву меншої потужності. Так можна досягнути повного заповнення головної частини і відповідно перейти на крок 5. Якщо отримуємо нестачу, то слід провести оцінку надлишку/нестачі елементів і застосувати процедуру віднімання.
5. Збереження результатів. Кінець.

Розглянемо процедуру оцінки надлишку/нестачі елементів, що використовується на кроці 4 попереднього алгоритму:

1. Послідовно вибирається коренева вершина зі списку, і якщо потужність відпо-

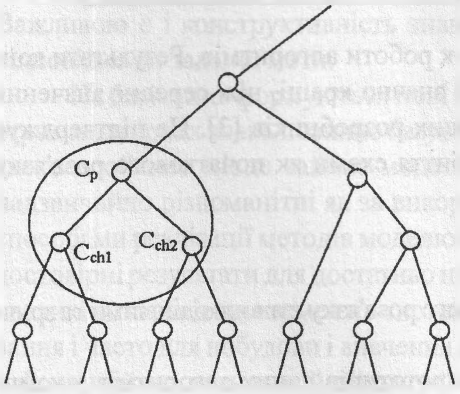
відного дерева менша ніж величина, на яку треба заповнити частину P_1, то всі елементи цього дерева призначаються в цю частину.

Якщо ще є дерева, менші за потужністю або ж рівні за нестачею, то повторюємо крок 1.

2. Визначаємо Δ_1 таким чином, що Δ_1 рівне незаповненій ємності частини P_1, тобто нестачі елементів у частині.
3. Визначаємо Δ_2 для найменшого непризначеного дерева, тоді $\Delta_2 = |T^R_i| - \Delta_1$. Розгляд проводиться, починаючи з найменшого дерева з метою зменшення обчислювальних затрат щодо перенесення кількості елементів.
4. Порівнюємо Δ_2 та Δ_1, щоб визначити, чи доцільно при поділі обраного дерева призначити його у доповнювальну частину P_2 та переносити з неї надлишок Δ_1 у головну при $\Delta_2 > \Delta_1$, чи навпаки – призначити в головну частину і переносити надлишок Δ_2 у доповнювальну частину. Цей крок проводиться з метою визначення меншої кількості елементів для переносу та мінімізації обчислювальних затрат.
5. Кінець.

Процедура віднімання надлишку базується на алгоритмі поділу дерева на дві частини P^T_1 і P^T_2 із заданою кількістю елементів [1]:

1. Визначаємо меншу за потужністю частину $P^T_{\min} = \min(|P^T_1|, |P^T_2|)$.
2. На обраному дереві T^R_i шукаємо кластер, який би містив кількість елементів найбільш наближену до кількості елементів меншої з частин, та мав найменшу кількість зовнішніх зв'язків.
3. Після вибору такого кластера задача зводиться до перенесення Δ елементів, на яку він більший або менший відносно частини $P^T_{\min}$.



Пошук базового кластера

Розглянемо детально крок 2 цієї процедури. Якщо розміри шуканого кластера значно менші ніж кількість елементів дерева, то ймовірно, що буде знайдено багато подібних за характеристиками кластерів і надалі слід серед них вибрати той, що найкраще задовольняє поставлені вимоги.

Для пошуку найкращого кластера на дереві згортання розглядаються групи з трьох кластерів – батьківський C_p та два його нащадки C_{ch1} та C_{ch2} (див. рис.), при умові, що $|C_p| \geq |P^T_{\min}| \geq |C_{ch1}|$ або/і $|C_p| \geq |P^T_{\min}| \geq |C_{ch2}|$. Розглядаються пари кластерів $\{C_p, C_{ch1}\}$, $\{C_p, C_{ch2}\}$. Для кожного кластера пари оцінюється відхилення Δ його потужності $|C_i|$ від потужності частини $|P^T_{\min}|$ та кількість зовнішніх зв'язків $|E^{ex}_{C_i}|$.

Критерій якості Q кластера визначається так:

$$Q = k_1 * \text{abs}(|C_i| - |P^T_{\min}|) + k_2 * |E^{ex}_{C_i}|$$

Можна гнучко керувати ваговими коефіцієнтами для знаходження кращого кластера, щоб мінімізувати кількість зовнішніх зв'язків та відхилення в бажаних межах.

Результати досліджень

Запропоновані алгоритми програмно реалізовано для застосування їх до схем великої розмірності. Протестовано реалізації програмного прототипу на кількох тестах фірми IBM [3]. Для першого тесту, що містить 12752 елементи, при розбитті цієї схеми на дві однакові частини, отримано результат щодо кількості зовнішніх зв'язків: 414. Дерево згортання для конструктивного розбиття будувалось згідно з такими вимогами:

1. Критерій згортання: максимізація різниці внутрішніх та зовнішніх зв'язків.
2. На кожному кроці відбувалось об'єднання незалежних пар елементів в кластери, відхилення значення критерію для яких не перевищувало 50% від максимального.

При виборі іншого критерію згортання – мінімізація кількості зовнішніх зв'язків – отримано результат: 375 зовнішніх зв'язків для кожної з частин.

Існуючі алгоритми розбиття застосовують серію спроб. Порівняємо отримані результати конструктивного розбиття з однією спробою із середнім значенням результатів роботи алгоритмів *FM*, *CLIP*, *hMetis* [4] за 100 спроб щодо поділу схеми на дві рівні частини.

У таблиці наведено мінімальне та середнє значення кількості зовнішніх зв'язків

Результати роботи алгоритмів при розбитті схеми на дві частини

Схема	FM		CLIP		hMetis		Конструктивне розбиття
	Мінімальне	Середнє	Мінімальне	Середнє	Мінімальне	Середнє	
Ibm01	191	466	181	390	181	236	375

для однієї частини схеми, що отримані внаслідок роботи алгоритмів. Результати конструктивного розбиття наближаються, а іноді й значно кращі, ніж середнє значення отримуваних результатів роботи алгоритмів інших розробників [3]. Це підтверджує доцільність застосування конструктивного розбиття схеми як початкового розв'язку для подальшого розбиття схеми.

Висновки

Запропоновані алгоритми пошуку початкового розв'язку є швидкодіючими та зручними для програмної реалізації.

Експериментальна програмна реалізація прототипів даних алгоритмів у комплексному пакеті для задач розбиття підтвердила їхню ефективність щодо пошуку якісних початкових розв'язків.

1. Базилевич Р.П. Декомпозиционные и топологические методы автоматизированного конструирования электронных устройств. Львов, 1981.
2. Базилевич Р.П., Подольський І.В. Особливості реалізації пакету програм для ієрархічної кластеризації схем// Вісник НУ"ЛП", 2002, № 440, стр. 139-144.
3. The ISPD98 Circuit Benchmark Suite Home Page, http://vlsicad.cs.ucla.edu/~cheese/ispd98.html
4. Charles J. Alpert, The ISPD98 Circuit Benchmark Suite. ISPD98 Monterey CA USA, 1998.