УДК 681.322

СЕРГІЄНКО А.М.

ПРОЕКТУВАННЯ ОБЧИСЛЮВАЧІВ З РЕГІСТРОВИМИ ЗАТРИМКАМИ

Пропонується методика проектування конвейерних обчислювачів, зконфігурованих в ПЛІС, яка забезпечує мінімізацію апаратурних витрат за рахунок широкого застосування регістрових затримок. Показана дієвість методики на прикладі проектування блоку тасування даних.

A method of designing pipelined datapaths which are configured in FPGA is proposed. The method provides the hardware minimization due to the wide utilization of the shift register components. The method is proven at the example of the zigzag scan reordering buffer design.

Програмовані логічні інтегральні схеми (ПЛІС) представляють собою незамінну елементну базу в таких галузях, як цифрова обробка сигналів, телекоммунікації, завдяки можливості організації в них швидкістного виконання алгоритмів в конвейерному режимі. Таке виконання підтримується архітектурою ПЛІС, яка включає велику множину конвейерних блоків множення з додаванням, пам’яті. Але розробка обчислювачів на базі ПЛІС залишається трудомістською, висококваліфікованою роботою. В статті пропонується методика проектування конвейерних обчислювачів, які в повній мірі використовують можливості апаратури сучасних ПЛІС.

В ПЛІС Virtex фірми Xilinx серед бібліотечних компонентів є елемент SRL16, що представляє собою 16-розрядний регістр зсуву з одним входом і одним виходом, який підключається до одного з тригерів цього регістру через багатовходовий мультиплексор, що керується чотирьохрозрядною шиною адреси. Цей елемент є регістровим буфером з регульованою затримкою або просто регістровою затримкою (РЗ) і призначений саме для організації конвейерних обчислень складних потокових алгоритмів. Ресурси РЗ досить великі – кожну другу логічну таблицю ПЛІС можна зконфігурувати як РЗ SRL16. Причому в критерії ефективності складність одного елемента SRL16 оцінюється як складність одного – двох тригерів. Крім того, одна РЗ може замінити собою до 16 регістрів, а також зекономити відповідні програмовані лінії зв′язку. Цим самим можна не тільки зменшити апаратні витрати, але і збільшити швидкодію обчислювача, зменшивши кількість ліній з′єднання з великою затримкою. Тому має сенс розробити методику еффективного використання ресурсів РЗ.

Загальний підхід до розробки функціональної схеми на рівні регістрових передач полягає в наступному. Вибирається множина ресурсів (суматорів, блоків множення, пам′яті і т.і.), складається розклад виконання операцій алгоритму і виконується призначення операцій на ресурси. За цим знаходять множину необхідних регістрів і мережу комутації ресурсів. Такий підхід також застосувють для проектування схеми з РЗ. При цьому ланцюжки регістрів, побудованих в схемі, заміняються на відповідні РЗ. Але ланцюжки регістрів в такій схемі виникають випадково і тому їх виявляється значно менше, ніж можливо і таким чином, РЗ в схемі задіяні нефективно. Крім того, властивість затримки РЗ, яка змінюється динамічно, не використовується.

В роботах [1–3] пропонується метод проектування конвейерних обчислювачів шляхом відображення графа синхронних потоків даних (ГСПД), який представлено в просторі ресурси – час у вигляді конфігурації алгоритму (КА). Метод дає змогу одночасно як складати розклад, мінімізувати кількість процесорних елементів (ПЕ), так і шукати ефективну систему з′єднань між цими ПЕ. Тут під ПЕ розуміється елементарний обчислювач з пам′яттю або без неї, наприклад, суматор, мультиплексор з регістром, РЗ, тощо. Тому має сенс створити методику розробки пристроїв з РЗ на основі цього методу.

На першому етапі синтезу за вказаним методом вершини-оператори однорідного ГСПД разом з дугами розташовуються в трьохвимірному просторі як множини векторів $\boldsymbol{K}_i$ та $\boldsymbol{D}_j$ з урахуванням умов, приведених в [2,3]. При цьому координати вектора $\boldsymbol{K}_i = (s,q,t)^T$ означають номер s ПЕ, де виконується оператор, тип q ПЕ і часову складову t, яка дорівнює номеру такту в періоді виконання алгоритму. Вектори

$\boldsymbol{K}_i$ з однаковою часовою складовою формують один ярус і тому виконуються одночасно. Часова складова $R(\boldsymbol{D}_j)$ вектора $\boldsymbol{D}_j = \boldsymbol{K}_i - \boldsymbol{K}_l$ дорівнює затримці між виконаннями операторів, вершини $\boldsymbol{K}_i$, $\boldsymbol{K}_l$ яких є суміжними. Виконується мінімізація числа ПЕ шляхом виконання вимог $|\boldsymbol{K}_{s,q}| \to L$, тобто число вершин, що відображаються в s-й ПЕ, прямує до L, де L – період виконання алгоритму, тактів. Крім того, при формуванні КА бажано будувати досконалий кістяк ГСПД, як це пропонується в [4].

На другому етапі виконується урівноважування КА, яке полягає в додаванні в дуги графа вершин затримки, поки часові складові усіх векторів $\boldsymbol{D}_j$ не дорівнюватимуть 0 або 1. Після цього виконується оптимізація КА шляхом взаємних перестановок векторів-вершин з одного ярусу з метою мінімізації числа регістрів та числа входів мультиплексорів в результуючій структурі і/або з застосуванням інших стратегій, наприклад, ресинхронізації [1]. Також мінімізується число регістрів склеюванням вершин затримки з одного яруса, які зберігають один і той самий операнд.

На третьому етапі одержана оптимізована КА відображається в граф структури обчислювача шляхом склеювання векторів-вершин з однаковими координатами s,q. КА перетворюється в розклад виконання операторів, використовуючи ту властивість, що часова складова вектору $\boldsymbol{K}_i$ дорівнює моменту виконання оператора безвідносно номера періоду виконання. При цьому можна не будувати структуру і розклад, якщо зразу описати схему обчислювача на мові VHDL [3].

Розглянемо деякий підграф КА, який виконується в РЗ. Цей підграф виконує пересилку операнда x з джерела $\boldsymbol{K}_{i1}$ до споживачів $\boldsymbol{K}_{j1}$, $\boldsymbol{K}_{l1}$ через дуги $\boldsymbol{D}_{j1} = \boldsymbol{K}_{j1} - \boldsymbol{K}_{i1}$, $\boldsymbol{D}_{l1} = \boldsymbol{K}_{l1} - \boldsymbol{K}_{i1}$, а також операнда y з джерела $\boldsymbol{K}_{i2}$ до споживачів $\boldsymbol{K}_{j2}$, $\boldsymbol{K}_{l2}$ через дуги $\boldsymbol{D}_{j2} = \boldsymbol{K}_{j2} - \boldsymbol{K}_{i2}$, $\boldsymbol{D}_{l2} = \boldsymbol{K}_{l2} - \boldsymbol{K}_{i2}$, відповідно (рис.1,а). Для того, щоб підграф виконувався на РЗ, принаймні всі його вхідні вершини повинні мати однакові просторові координати p.

При виконанні алгоритму в РЗ операнди x і y в кожному такті пересилаються на наступний регістр РЗ. Це еквівалентно тому, що в урівноваженій КА ці операнди в кожному такті передаються в наступний ярус і наступний ряд вершин затримки, який відображається в регістр РЗ. Іншими словами, ланцюжки суміжних вершин затримки $\boldsymbol{K}_{Di}$ при рівномірно зростаючих координатах $R(\boldsymbol{K}_{Di})$ розміщуються вздовж паралельних прямих, розташованих під одним кутом до осі ot. Пройшовши ланцюжки з $R(\boldsymbol{D}_{j1})$, $R(\boldsymbol{D}_{l1})$, $R(\boldsymbol{D}_{j2})$, $R(\boldsymbol{D}_{l2})$ вершин затримки, операнди x і y видаються на відповідні вихідні вершини підграфу (рис.1,б). Результуюча схема РЗ показана на рис.1,в.

РЗ має єдиний вихід, тому на підграф КА накладається додаткове обмеження – з вершин затримки, які належать одному ярусу і які відображаються в РЗ, повинно виходити не більше ніж одна дуга, що веде у вихідну вершину. При цій умові вихідний мультиплексор РЗ одночасно підключатиметься тільки до одного регістра РЗ. В іншому разі РЗ повинен мати більше, ніж один вихід або вихідний мультиплексор, як показано пунктиром на рис.1,б,в. Такий підграф повинен бути реалізований на кількох РЗ, як на рис.1,г.

РЗ типу SRL16 має додатковий вхід дозволу синхросерії, керування яким дає змогу загальмувати просування операндів по регістрах РЗ. При використанні цього входу можна зекономити кількість РЗ, якщо величина $R(\boldsymbol{D}_j)$ більше кількості регістрів в РЗ. На рис.2 показано приклад перетворення КА на рис.1,б з метою додаткової затримки на такт операндів, які поступають в вершини $\boldsymbol{K}_{l1}$, $\boldsymbol{K}_{l2}$. Така затримка відповідає векторам $\boldsymbol{D}_j$, які розміщуються паралельно осі ot.

Якщо вершини-джерела операндів мають різні просторові координати s, то на вході РЗ одержується вхідний мультиплексор. Мінімізацію входів таких мультиплексорів можна виконувати згідно з методикою, приведеною в [5].

Таким чином, методика проектування конвейерних обчислювачів з РЗ виглядає як наступна. Начальні дані – КА, період виконання алгоритму L та інші параметри оптимізації. Методика виконується таким самим чином, як це описано в [1], за виключеннями, які описані нижче.

На першому етапі синтезу слід виділити підграфи КА, що відповідають пересилці операндів між ресурсами обчислювача з затримками у часі і/або тасуванням операндів, які передбачається відобразити в окремі РЗ.

На другому етапі треба урівноважити дуги залежностей за допомогою проміжних вершин затримки. Виконати зменшення кількості проміжних вершин затримки для всіх дуг, якщо можливо.

Розмістити вершини затримки на паралельних прямих, що знаходяться під кутом до осі часу або паралельно цій осі таким чином, щоб суміжні вершини затримки відрізнялись по часовій координаті на один такт. Виконати вимоги коректного розміщення вершин, включаючи вимогу реалізації РЗ з одним входом і виходом. У разі неможливості одержати один вхід у РЗ використовують евристику мінімізації числа входів додаткового мультиплексора на вході РЗ згідно з [5], а при неможливості одержати РЗ з одним виходом ланцюжки вершин затримки розщеплюють, щоб вони відображались в додаткові РЗ (див. рис.1,г).

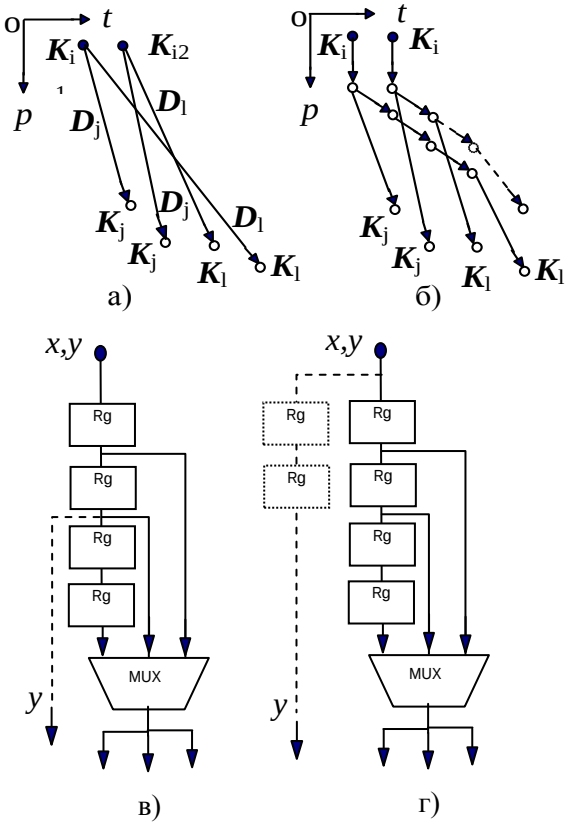

Рис.1. Відображення підграфу КА в РЗ

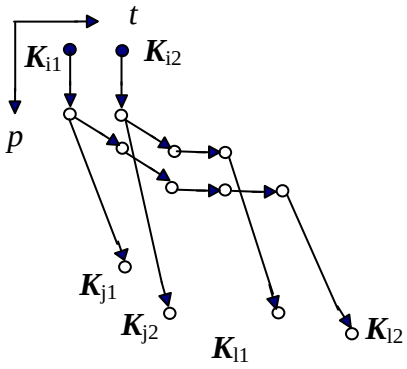

Рис.2. КА, що відповідає РЗ з входом дозволу

Дуги залежностей разом з відповідними вершинами затримки, що інцидентні вершинам-приймачам, відобразити в РЗ. При складанні алгоритму керування обчислювачем якщо в РЗ відображаються тільки дуги, які знаходяться під кутом до осі часу, то операнди в РЗ записуються в кожному такті, а якщо є дуги паралельні цій осі, то у відповідних їм тактах забороняється запис в РЗ.

На третьому етапі обчислювач, описаний на мові VHDL згідно з методикою, представленою в [1], компілюється в конфігурацію ПЛІС, яка вміщує РЗ типу SRL16, які відповідають виділеним підграфам КА.

Розглянемо приклад проектування буфера тасування даних згідно з правилом z-видного обходу, який використовується в кодерах зображення за стандартом H264 [6]. Якщо 16 вхідних даних приходять на вхід такого буфера в натуральному послідовному порядку, то вони виходять з нього в порядку згідно з послідовністю: 0,1,4,8,5,2,3,6,9,12,13,10,7,11,14,15. Як правило, такий буфер будують на основі двохпортової оперативної пам′яті, що призводить до нераціонального використання ресурсів, а також до великої латентної затримки між прийомом вхідних даних і видачею вихідних даних.

Урівноважена КА алгоритму роботи такого буфера, яка підготовлена згідно з одержаною методикою, показана на рис.3. Опис КА на мові VHDL приведено нижче.

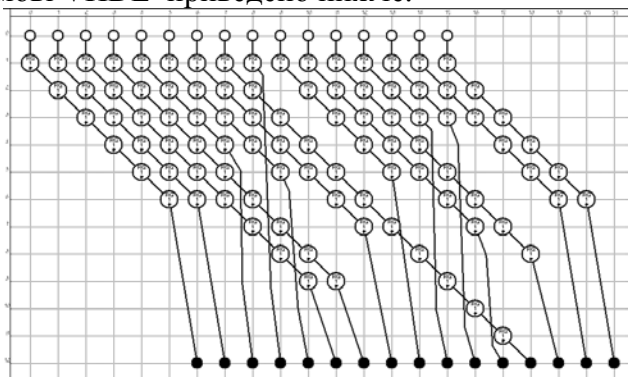

Рис.3. Урівноважена КА буфера тасування

Одержаний буфер при конфігуруванні в ПЛІС Virtex4 фірми Xilinx має мінімальні апаратні витрати – 4 тригери, 9 логічних таблиць і 12 елементів SRL16 – за кількістю розрядів даних. Цей буфер може бути використаний при рекордній тактовій частоті 900 МГц. Такі результати показують дієвість розробленої методики.

Було розроблено ряд конвейерних процесорів швидкого перетворення Фур′є з алгоритмом Винограда. Цей алгоритм характеризується складними перестановками даних і тому мінімальні апаратурні витрати процесорів досягнуті саме завдяки використанню запропонованої методики.

```
entity ZZ4x4 is
   port(CLK : in STD_LOGIC;     – синхросигнал
      START : in STD_LOGIC;     – запуск буфера
      DI : in STD_LOGIC_VECTOR(11 downto 0); – вхідні дані
      DO : out STD_LOGIC_VECTOR(11 downto 0)   ); – вихідні дані
end ZZ4x4;
architecture ZZ4x4 of ZZ4x4 is
   type TARR16 is array (0 to 15) of bit_vector(11 downto 0);
   type TA is array(0 to 15) of natural range 0 to 10;
   signal sr:TARR16;                                  – масив регістрів SRL16
   constant table :TA:=(5,5,3,0,4,8,8,6,4,2,2,6,10,7,5,5); – номери відводів SRL16
   signal fa, addr:natural range 0 to 15;
begin
   FSM:process(CLK,RST)  begin      – лічильник тактів  періоду алгоритму
      if CLK'event and CLK='1' then
         if START='1' then  addr<=0;  else addr<=(addr+1) mod 16; end if;
      end if;
   end process;
   fa<=table(addr);        – перекодування такту в номер відводу SRL16
   SRL16:process(CLK)  begin                  – опис SRL16
      if CLK'event and CLK='1' then
         sr<=DI & sr(0 to 14);                – власне зсув в РЗ
      end if;
   end process;
   DO<= sr(fa);              – видача результату з fa-го відводу  SRL16
end ZZ4x4;
```

Таким чином, запропонована методика проектування конвейерних обчислювачів, зконфігурованих в ПЛІС, яка забезпечує мінімізацію апаратурних витрат за рахунок широкого застосування регістрових затримок. Методика може бути використана для проектування спеціалізованих обчислювачів, в яких використовується буферна пам′ять типу FIFO.

Перелік посилань

1. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – Киев: ДиаСофт. – 2003. – 203 с.
2. Сергієнко А.М. Синтез структур для виконання періодичних алгоритмів з операторами керування // Вістник НТУУ “КПІ”. Сер. Інформатика, управління та обчислювальна техніка. – 2007. – №47. – с.221–227.
3. Каневский Ю.С., Овраменко С.Г., Сергиенко А.М. Отображение регулярных алгоритмов в структуры специализированных процессоров // Электрон. Моделирование.–2002.–Т.24.–№2.–С. 46-59.
4. Сергієнко А.М. Досконалий кістяк алгоритму.
5. Сергиенко А.М., Симоненко В.П. Отображение периодических алгоритмов в программируемые логические интегральные схемы //Электронное моделирование. –Т.29. –2007.–№2.–с.49–62.
6. Richardson I.E.G. H.264 and MPEG-4 Video Compression. Video Coding for Next-generation Multimedia. –Wiley. –2003. –281p.