УДК 004.315.5

Лісовий О.М.
Опанасенко В.М., д-р техн. наук
Сорока Є.В.

РЕАЛІЗАЦІЯ УНІВЕРСАЛЬНИХ КОМПОНЕНТІВ НА ПЛІС

Інститут кібернетики ім. В.М. Глушкова НАН України

Розроблено універсальні компоненти перетворення форматів представлення чисел на базі ПЛІС типу FPGA. Використовується поведінковий та змішаний опис конвеєрної архітектури компонентів мовою VHDL. Проведена перевірка функціонування компонентів методом моделювання в системі ModelSim Xilinx Edition - MXE III за допомогою поведінкового стенду та файлу вхідних впливів

Вступ

Якщо донедавна основним способом опису проектів був графічний синтез схеми пристрою, який розроблявся схематичним редактором, то останнім часом перевага віддається текстовому опису, подібному до тексту програми.

Опис у вигляді тексту на *HDL*-мові регламентується відповідними стандартами. Слід зазначити, що синтаксично правильний опис на *HDL*-мові не завжди може бути синтезовано, тобто представлено у вигляді апаратної моделі, отриманої після компіляції з *HDL*-коду. Так, наприклад, не можуть бути синтезовані сигнали й змінні типу із плаваючою точкою, оператори *Wait* з параметром затримки й т.д. Перелік несинтезованих елементів багато в чому залежить від конкретного засобу синтезу й типу обраного кристала. Деякі конструкції можуть синтезуватися в різних проектах по різному, приміром, конструкція *For* може синтезуватися як послідовна комбінаційна схема (ітерації виконуються послідовно за допомогою одного синхросигнала) або як схема, кожна ітерація циклу буде виконуватися по новому синхросигналу.

В мові *VHDL* існує декілька способів опису архітектури компонентів [1, 2]: поведінковий – опис вхідних/вихідних портів компоненту та формування даних на вихідних портах в залежності від вхідних даних (алгоритмічний опис поведінки компоненту) при чому внутрішня структура об'єкту що описується не специфікується; структурний – опис структури компоненту, який складається з інших компонентів, включає перелік цих компонентів та зв'язки між ними, компоненти описуються окремо та їх опис може бути поведінковий, структурний або змішаний; змішаний – структурно-поведінковий опис є комбінацією структурного та поведінкового.

Вибір опису архітектури компоненту відбувається в залежності від формальної декомпозиції алгоритму цього компоненту, як правило для опису нескладних компонентів використовується поведінковий опис, а для опису складних або тих, що використовують раніше розроблені компоненти, структурний або змішаний. Зауважимо що способи опису архітектури компоненту не є взаємовиключаючі, тобто складні компоненти (конвеєри, машини станів) можуть бути описані довільним описом. На прикладі компонентів що реалізують перетворення форматів розглянемо конвеєр (перетворювач ФПТ [3] (*Single, Double, Extended, Quadruple*) в ФФТ *(Integer, Int64, Int128)*) з поведінковим описом архітектури, та конвеєр з змішаним описом архітектури (перетворювач ФФТ в ФПТ), що відрізняється від структурного лише деякою кількістю допоміжних регістрів (процес *make_RG*) та нескладною комбінаційною логікою (процес *make_oDat*).

Постановка задачі

Перетворення форматів [4] є невід'ємною складовою набору операцій над числами в ФПТ [5, 6]. Серед *IP-Core* фірми *Xilinx* для операцій в ФПТ є опера-

ції перетворення максимальна розрядність ФПТ дорівнює 80 біт. Дані в ФФТ 64 біт та до 32 біт для дрібної частини (при перетворені ФПТ-ФФТ є можливість зберегти дрібну складову дійсного числа.

Розглянемо умовно-графічне позначення (УГП) та вхідні/вихідні порти компонентів. *«i»* – *input* (вхідні), *«o»* – *output* (вихідні), *«fp»* – *float point* (ФПТ), *«fixp»* – *fix point* (ФФТ) *«Dat»* – дане, *«CLK»* – синхросигнал, *«PK»* – признак доповняльного коду, *«RDY»* – *ready* (сигнал готовності вихідних даних). Не зважаючи на те, що вхідні і майже всі вихідні порти компонентів майже однакові вони виконують різну функцію відповідно розглядатимемо їх окремо.

Перетворювач ФФТ-ФПТ – конвеєр з чотирьох ступенів, структурна схема компоненту наведена на рис. 2 а. Де *«PK»* вказує в якому коді подається вхідне дане (*«0»* – в прямому коді, *«1»* – в доповняльному коді); *«ifixp»* – вказує в якому форматі подається вхідне дане, та є сигналом загрузки даних в конвеєр (початком роботи) (*«00»* – стан очікування даних, *«01»* – дане в форматі *Integer*, *«10»* – дане в форматі *Int64*, *«11»* – дане в форматі *Int128*); *«ifp»* – вказує в який ФПТ потрібно перетворити дане (*«000»* – не встановлено користувачем, оскільки діапазон допустимих значень *Single* перекриває діапазони всіх ФФТ то перетворення відбуватиметься в *Single*, *«001»* – перетворити в *Single*, *«010»* – в *Double*, *«011»* – в *Extended*, *«100»* – в *Quadruple*); *«ofp»* кодує формат вихідного даного відповідно кодуванню *«ifp»*, вихідне дане вважається отриманим, коли присутній сигнал *«RDY=1»* та *«ofp≠000»*, коли *«RDY=1 ofp=000»* результат машинний нуль, коли *«RDY=0»* – режим очікування або завантаження конвеєру.

Перетворювач ФПТ-ФФТ – конвеєр з трьох ступенів, структурна схема компоненту наведена на рис. 2 б. Де *«PK»* вказує в якому коді формувати вихідне дане (*«0»* – в прямому коді, *«1»* – в доповняльному коді); *«ifixp»* – вказує в якому форматі формувати вихідне дане (*«00»* – не встановлено користувачем, намагається виконати перетворення в формат *Integer*, якщо неможливо то в *Int64*, якщо неможливо то в *Int128*, якщо неможливо то перетворення неможливе, *«01»* – в *Integer*, *«10»* – в *Int64*, *«11»* – в *Int128*); *«ifp»* – вказує в якому ФПТ подається вхідне дане, та є сигналом загрузки даних в конвеєр (початком роботи) (*«000»* – стан очікування даних, *«001» – Single, «010» – Double, «011» – Extended, «100» – Quadruple); «ofixp»* кодує формат вихідного даного відповідно кодуванню *«ifixp»*, вихідне дане вважається отриманим, коли присутній сигнал *«RDY=1»* та *«ofixp≠00»*, коли *«RDY=1 ofixp=00»* результат машинний нуль або неможливо виконати перетворення (в зв'язку з значно меншим діапазоном ФФТ від ФПТ), коли *«RDY=0»* – режим очікування або завантаження конвеєру.

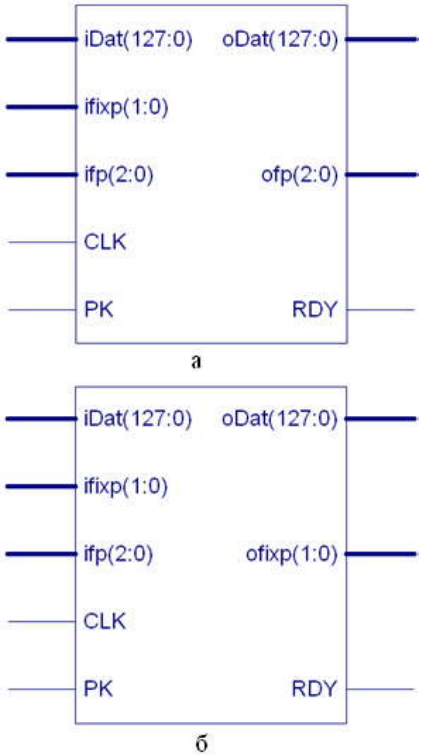

Рис. 1.
а) УГП перетворювача ФФТ-ФПТ;
б) УГП перетворювача ФПТ-ФФТ

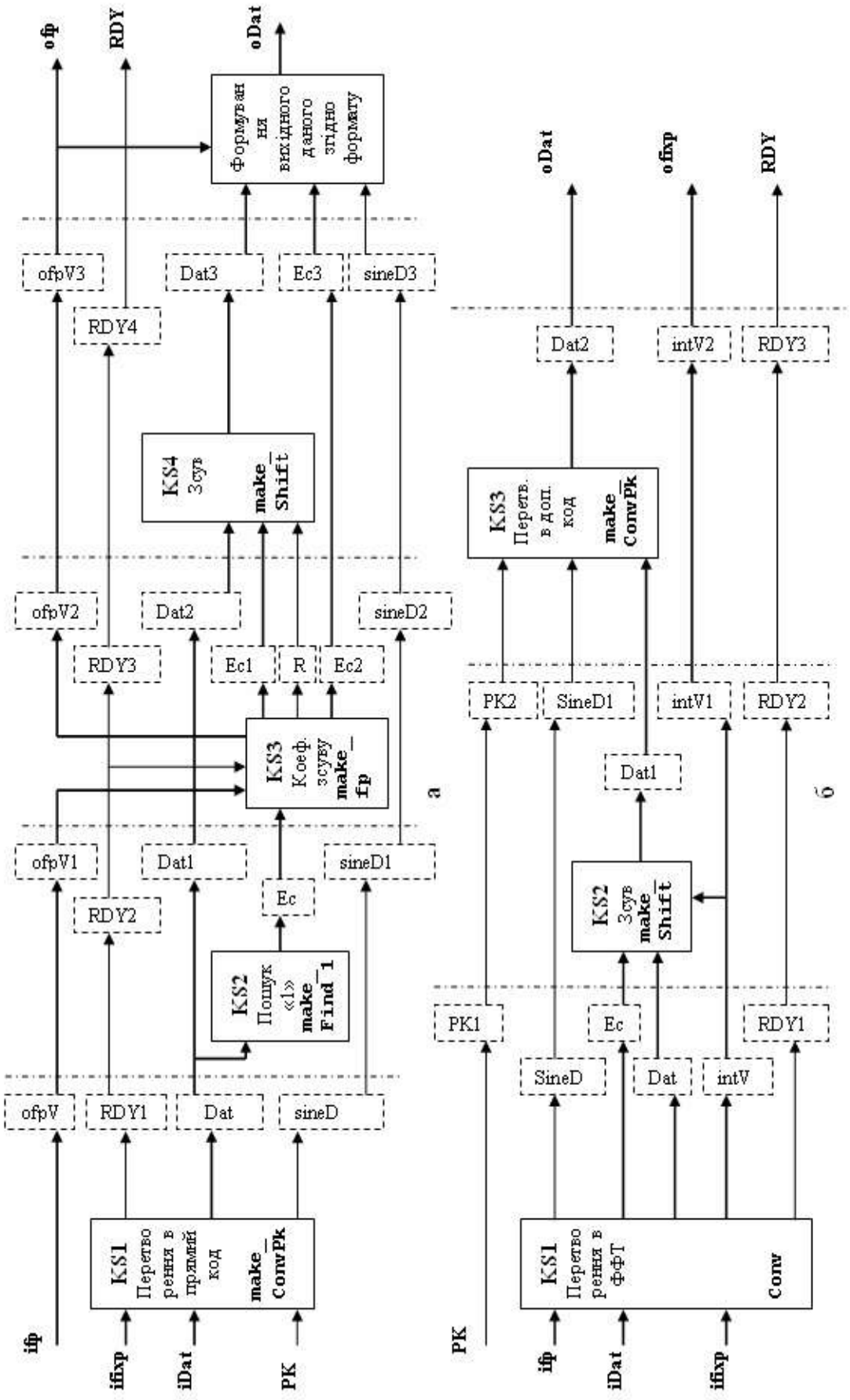

Рис. 2. Структурна схема перетворювача: а) ФФТ-ФПТ; б) ФПТ-ФФТ

Дані в ФПТ та ФФТ в портах *Dat* (вхідні та вихідні) відповідають наступним розрядам: *Single* в *Dat(31:0), Double* в *Dat(63:0), Extended* в *Dat(79:0), Quadruple* займає всю розрядність порту, *Integer* в *Dat(31:0), Int64* в *Dat(63:0), Int128* займає всю розрядність порту.

Розглянемо структурні схеми перетворювачів ФФТ-ФПТ на рис. 2. а ФПТ-ФФТ на рис. 2. б. Штрих пунктиром розділені ступені конвеєрів, блоки з штрихових ліній – регістри (всі синхронні) інші блоки комбінаційні схеми. Назви регістрів співпадають з назвами сигналів в описі архітектури компонентів, та в нижній частині комбінаційних схем зазначені назви процесів в яких реалізована відповідна комбінаційна логіка.

Реалізація перетворювачів

Поведінковий опис конвеєру ФПТ-ФФТ складається з трьох процесів які описують алгоритм операції перетворення та процесу який формує зв'язки між регістрами (*make_RG*). Процес *Conv* виконує головну функцію в перетворені, а саме, формування попереднього вектору (*Dat*) та числа на яке потрібно цей вектор зсунути вправо (*Ec*). На формування вектору (*Dat*) впливають сигнали *ifp, ifixp, iDat,* які також формують сигнал *oIntV*, значення якого вказують на формат вихідного даного, кодування відповідає *ofixp*. Процес *make_Shift* формує вектор (*Dat1*) виконуючи зсув вправо вектору (*Dat*) на (*Ec*). Процес *make_ConvPk* в залежності від сигналу *PK2* формує кінцевий результат в прямому або доповняльному коді.

Змішаний опис конвеєру ФФТ-ФПТ окрім опису портів чотирьох компонентів *KS1..KS4* та взаємозв'язку між ними має два процеси: *make_oDat* – формування кінцевого результату (без регістра на виході) та *make_RG* який формує зв'язки між регістрами. Компонент *KS1* – виконує формування попереднього вектору (*Dat*) відповідно до значень *PK, ifixp, iDat*. Компонент *KS2* – пошук першої старшої одиниці (*Ec* – її індекс в векторі). Компонент *KS3* – формування коефіцієнтів *Ec1* – число на яке потрібно виконати зсув вектору, *Ec2* – ступінь числа з плаваючою точкою (сформовано відповідно до вибраного ФПТ), *R* – біт зсуву (*«0»* - вправо, *«1»* – вліво), формується покажчик ФПТ, вразі некоректно заданого (*ifp*) коду користувачем формується покажчик помилки (*«000»*). Компонент *KS3* – виконує зсув вправо/вліво (залежно від *R*) вхідний вектор.

В таблиці наведено дані про апаратні ресурси та максимальній швидкодії розроблених компонентів.

Таблиця. Апаратно-часові характеристики для мікросхеми ПЛІС *3s1500fg676-5*

Компонент	Кількість ступенів	Кількість			Період синхросигналу *ns*
		Slices	*FF*	*LUT*	
Universal_conv_FP_FixP	3	855(6%)	421(1%)	1470(5%)	17.075
Universal_conv_FixP_FP	4	1680(12%)	652(2%)	2962(11%)	15.189

Верифікація компонентів

Верифікація компонентів виконується за допомогою системи моделювання *ModelSim XE.* Формування вхідних сигналів виконується в вигляді *VHDL*-файлу вхідних впливів.

Верифікація компонента перетворювача ФФТ-ФПТ. Вхідні дані з 1 по 14, подаються з відміткою *Integer*, дані з 15 по 20 в *Int64*, дані з 21 по 26 в *Int128*, перетворюються в результат з 1 по 26 в всі ФПТ. Оскільки діапазон представлення чисел в всіх ФФТ менше ніж діапазон найменшого ФПТ (*Single*), то втрати значимості при перетворені не буде в жодному з випадків, однак може відбутися втрата значимих бітів (двійковий порядок числа коректний, похибка дорівнює вектору відсутніх молодших бітів, який не помістився в мантисі). Розглянемо випад-

ки коли може відбутися втрата значущих бітів: перетворення з *Integer* в *Single*, коли в вхідному векторі є значущі біти в (30..24), з Integer в інші ФПТ втрат немає; з *Int64* в *Single*, коли в вхідному векторі є значущі біти в (62..24), з *Int64* в *Double*, коли в вхідному векторі є значущі біти в (62..53), з *Int64* в інші ФПТ втрат немає; з *Int128* в *Single*, коли в вхідному векторі є значущі біти в (126..24), з *Int128* в *Double*, коли в вхідному векторі є значущі біти в (126..53), з *Int128* в *Extended*, коли в вхідному векторі є значущі біти в (126..64), з *Int128* в *Quadruple*, коли в вхідному векторі є значущі біти в (126..113).

Верифікація компонента перетворювача ФПТ-ФФТ. Вхідні дані з 1 по 7, подаються з відміткою *Single*, дані з 8 по 14 в *Double*, дані з 15 по 20 в *Extended*, дані з 21 по 26 в *Quadruple* перетворюються в результат з 1 по 26 в всі ФФТ. Діапазон представлення чисел в ФФТ менше ніж в ФПТ а тому в деяких випадках відбуватиметься втрата значимості. Оскільки в ФПТ використовується для дійсних чисел то після перетворення отримаємо лише цілу складову цього числа.

Висновки

Використання широкоформатних обчислень з плаваючою точкою (*Quadruple*) – один із актуальних механізмів розв'язання задач з оцінкою достовірності та малою похибкою обчислень. Сьогодні 128 бітну апаратну підтримку мають процесори *AMD Phenom* з підтримкою *FPU* (*floating point unit*) 128 біт та процесори *SPARC-V9 Sun* з підтримкою *DFPU* (*decimal floating point unit*) 128 біт.

Компоненти перетворення форматів розроблено за допомогою інструментальних засобів *Xilinx Foundation ISE 9.2* та перевірено на етапах розробки окремих блоків, а також компонентів в цілому. Для тестування використано систему моделювання *ModelSim XE3* 6,3 яка дозволяє окрім верифікації встановити апаратно-часові характеристики пристроїв, які представлено в таблиці. Перевагою проектування арифметичних пристроїв на ПЛІС є можливість модернізації алгоритмів обчислень та збільшення розрядності оброблювальних даних.

Розроблені компоненти можуть використовуватися, як елемент зв'язку, в складних обчислювальних пристроях для реалізації обчислень з фіксованою та плаваючою точкою. Також дані компоненти можуть бути корисними для тих математичних задач, в процесі рішення яких в ФФТ відбувається втрата значимості (через невеликий діапазон, по відношенню до ФПТ, представлення чисел), дані можуть бути конвертовані в ФПТ, розв'язані в ФПТ, та конвертовані в ФФТ, як кінцевий результат обчислень.

Список літератури

1. *Семенец В.В., Хаханова И.В., Хаханов В.И.* Проектирование цифровых систем с использованием языка *VHDL*. – Харьков: ХНУРЕ, 2003. – 492 с.

2. *Суворова Е.А., Шейнин. Ю.Е.* Проектирование цифровых систем на *VHDL* – СПб.: БХВ-Петербург, 2003. – 576 с.

3. *Hollash S. IEEE Standard 754 Floating Point Number / S. Hollash / Available at http://IEEE/ IEEE Standard 754 Floating Point.html.*

4. *Опанасенко В.Н., Лисовый А.Н.* Особенности языка *VHDL* для программирования кристаллов ПЛИС. – К.: Проблеми програмування. – №1. – 2006. – 102 с.

5. *Опанасенко В.Н., Сахарин В.Г., Лисовый А.Н.* Проектирование модулей с плавающей точкой на ПЛИС с использованием языка *VHDL*. – К.: Математические машины и системы. – №3. – 2005. – 195 с.

6. *Опанасенко В.Н., Сахарин В.Г., Лисовый А.Н.* Использование *VHDL*–технологии при проектировании конвейерных устройств на базе ПЛИС // Вестник НТУ «ХПИ»: Сб. научных трудов. – №55. – Харьков. – 2005. – С. 151–15.