УДК: 004.315.8
Л.О.Цигилик
Національний університет "Львівська політехніка»

ПРОЕКТУВАННЯ КОНФІГУРОВАНОГО ПРИСТРОЮ ДЛЯ ВИКОНАННЯ ЛОГІЧНИХ ОПЕРАЦІЙ

У роботі запропоновано рекомендації щодо проектування засобів генерації та конфігурування логічних пристроїв. Описано методологію проектування конфігурованої системи виконання логічних операцій. Подано результати верифікації системи для двох наборів конфігураційних параметрів, проведено аналіз і визначено переваги та недоліки.
Ключові слова: *конфігурація системи, логічні операції, типи даних, комбінаційна схема.*

ВСТУП

Поява ПЛІС відкрила широкі можливості для проектування систем на кристалі. Використовуючи апаратну базу та відповідний САПР ПЛІС, а також готові технічні рішення, так звані IP ядра (Intellectual Property) [1], можна реалізувати доволі складний проект за короткий термін. Перевагою використання IP ядер є гнучкіть в проектуванні системи. Інженер вибирає параметри налаштування ядра згідно з вимогами вирішуваної задачі, іншими словами виконує конфігурацію системи.

1. ОГЛЯД ЛІТЕРАТУРИ

Конфігурування як підхід використовується також у проектуванні арифметико-логічних пристроїв [5]. Беручи моделі вже існуючих IP ядер (програмне забезпечення для генерації VHDL коду) у налаштуваннях програми інженер вказує розрядність шини даних, множину арифметичних і логічних операцій та тип представлення даних (цілочисельний тип, формат з фіксованою чи рухомою комою). Змінюючи той чи інший елемент налаштування створюється новий модуль системи із мінімальними затратами часу на його проектування.

Використання комбінаційних схем в проектуванні має ряд переваг в порівнянні з послідовними [4,5]. В таких схемах відсутні елементи пам'яті, що не створюють додаткових затримок реагуючи на зміну сигналу запису/читання. Сигнал на виході комбінаційної схеми в момент часу $_{\Delta}t+t$, прямо залежить від сигналу на вході схеми в момент часу t. $_{\Delta}t$ - сумарний час затримки, яку створюють логічні вентилі та зв'язки між ними в комбінаційній схемі.

В даній статті наведено рекомендації щодо проектування конфігурованого модуля для виконання логічних операцій над цілочисельним типом даних на основі комбінаційної схеми.

2. ЗАСОБИ ТА МЕТА РОБОТИ

Метою роботи є формування основних етапів проектування програмного засобу (надалі генератора коду), завданням якого є виконання автоматичної генерації VHDL коду. Даний код описує апаратний пристрій, який виконує логічні операції над цілочисельним типом даних. Вимоги до генератора наступні:

1. Можливість конфігурування. У налаштуваннях програми потрібно забезпечити можливість вказувати розрядність шини даних та перелік логічних операцій.
2. Так як в логічному пристрої відсутні елементи пам'яті, доцільно виконати його синтез на основі комбінаційної схеми.
3. Логічний пристрій повинен виконувати наступний перелік операцій:
   - not – логічне заперечення;
   - and – логічне «і»;
   - or – логічне «або»;
   - xor – виключне «або»;
   - eq – рівність;
   - neq – нерівність;

- gr – операція « > »;
- greq – операція « >= »;
- les – операція « < »;
- leseq – операція « <= »;
- sll – логічний зсув вліво, вільні розряди заповнюються «0»;
- srl – логічний зсув вправо, вільні розряди заповнюються «0»;
- sla – арифметичний зсув вліво, вільні розряди заповнюються «1»;
- sra – арифметичний зсув вправо, вільні розряди заповнюються «1»;
- rol – циклічний зсув вліво, при зсуві старших розрядів вони заповнюють місце молодших;
- ror – циклічний зсув вправо, при зсуві молодших розрядів вони заповнюють місце старших.

Засобами для досягнення поставленої мети є пакет Microsoft Visual Studio 2010 для створення генератора коду, Active HDL 8.3 для верифікації системи та Xilinx ISE 10.1 [3] для синтезу кристалу та оцінки апаратних витрат.

3. МЕТОДИКА ПРОЕКТУВАННЯ

Процедура проектування логічного пристрою складається з двох етапів:

1. Опису на мові VHDL логічного пристрою, який виконує усі вище перелічені операції та його верифікація.
2. Створення генератора VHDL коду на мові високого рівня. Підготовка множин тестових даних, для кожної окремої множини даних генерація VHDL коду та його верифікація.

Проектуючи логічний пристрій на основі комбінаційної схеми виділяємо основні вузли системи. На рис. 1 показано функціональну схему логічного пристрою, яка складається з вузлів зсуву (sll, srl, sla, sra, rol, ror), чотирьох вузлів виконання булевих операцій (not, and, or, xor), вузла ch_shift який призначений для зміни другого операнда та зміни коду виконання логічної операції, вузла relation_ch_res, призначеного для реалізації операцій відношення (==, /=, > ,< , >= , <=) та вибору результату.

Формат цілочисельного типу даних. Додатні цілі числа, представленні в прямому коді [2] – напр. для даних розрядністю 12 біт: $26_{10} = 000000011010_2$, від'ємні числа, представлені в доповняльному коді, який утворюється шляхом інвертування числа, представленого в прямому коді та додаванням «1» напр.: $-26_{10} = 111111100110_2$.

Алгоритмом виконання операцій логічних зсувів передбачено, що якщо другий операнд (рис. 1 операнд B) від'ємний, то операція зсуву відбувається в протилежному напрямку на ту кількість розрядів, значення якої має модуль операнда B напр.:

$$A: 12_{10} = 01011_2 \; ; \qquad B: -3_{10} = 11101_2 \; ; \qquad C: 3_{10} = 00011_2 \; ;$$

$$S: \; sll \rightarrow "01"; \qquad srl \rightarrow "10"$$

$$S = "01"; Q = sll \rightarrow A, B \qquad \Leftrightarrow \qquad S = "10"; Q = srl \rightarrow A, C$$

$$Q = 00001_2$$

Для виконання даного кроку алгоритму розроблено модуль ch_shift, який виконує наступну функцію: якщо код операції рівний одній з операцій логічних зсувів та число B від'ємне – виконується процедура перетворення числа з доповняльного коду в прямий, шляхом віднімання «1» та інвертування числа, і заміни коду операції із зсуву операнда вліво на зсув вправо та навпаки. Якщо число B додатнє – ніяких змін не відбувається.

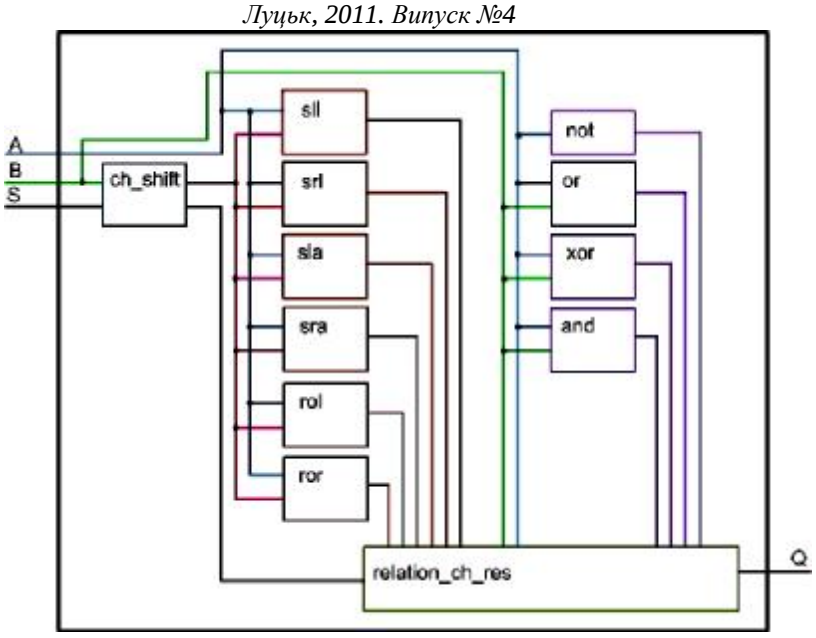

Рис. 1. Функціональна схема пристрою виконання логічних операцій

Генератор VHDL коду має наступну інформаційну структуру ((1):

$$GEN = \{r\_data, r\_op, COMAND\} \qquad (1)$$

де, r_data - розрядність шини даних, r_op - розрядність коду операції та множину $COMAND = \{op, code\_op\}$ в якій op - символьне представлення логічної операції, а $code\_op$ - відповідний до неї двійковий код операції. Дана інформаційна структура заповнюється в налаштуваннях програми генератора.

Описуючи генератор потрібно звернути увагу на незалежність окремих конфігурованих частин VHDL коду. Це означає, що якщо в елементі op присутня та чи інша команда, то всі зв'язки (сигнали, операції, модулі) повинні генеруватися у VHDL код незалежно один від одного. Якщо однієї чи декількох команд немає у вхідних налаштуваннях генератора, то при створені VHDL коду та його компіляції не повинно виникати конфліктних ситуацій та помилок. Структура генератора повинна мати структуру шаблону. Аналізуючи налаштування програми в шаблон підставляється розрядність шини даних, згідно з даною розрядністю генерується той чи інший вузол на виконання логічної операції.

Підсумовуючи, можна сказати що процес проектування системи за допомогою конфігурування існуючих IP ядер є перспективним напрямком, який інтенсивно розвивається. Основною перевагою є те, що у конфігурованій системі немає надлишковості. Генеруються лише необхідні вузли системи згідно налаштувань генератора VHDL коду.

4. ВЕРИФІКАЦІЯ КОНФІГУРОВАНОГО МОДУЛЯ

Перевірка функціональності роботи системи відбувалася в пакеті Active HDL 8.3. Для генерації VHDL коду було вибрано такі налаштування системи, конфігурація №1: список восьми довільних логічних операцій, розрядність шини даних складає 24 біти, розрядність коду команди – 4 біти.

На рис. 2 показано часову діаграму роботи системи, а у лівій стороні – список логічних операцій. Тут вибрано лише частину логічних операцій, для того щоб показати як впливає кількість та складність операцій на витрату апаратних ресурсів. Дослідження затрат обладнання було виконано за допомогою пакету Xilinx ISE 10.1 для ПЛІС Virtex 4 FX100 [3].

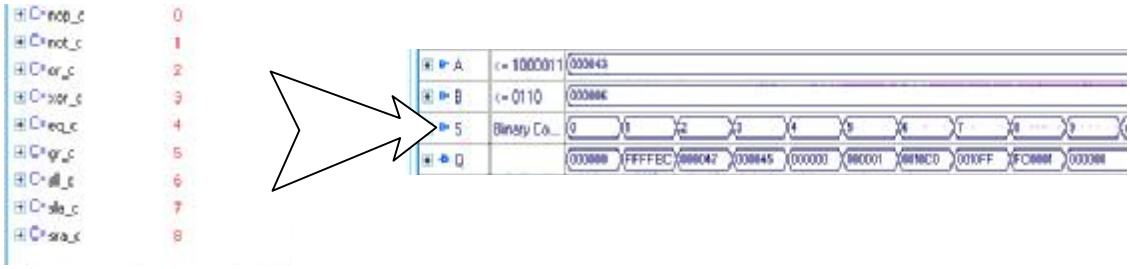

Рис. 2. Список логічних операцій та часова діаграма роботи системи №1.

Створимо нову систему змінивши налаштування генератора на такі, конфігурація №2: список дев'яти довільних логічних операцій, розрядність шини даних складає 13 біт, розрядність коду команди – 4 біти. На рис. 3 показано часову діаграму роботи системи згідно конфігурації №2.

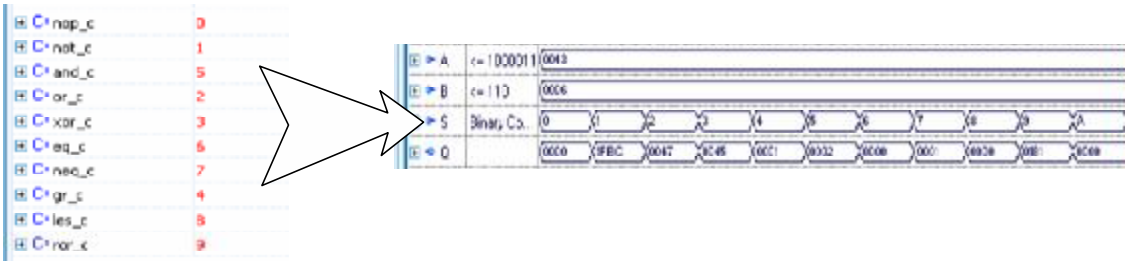

Рис. 3. Список логічних операцій та часова діаграма роботи системи №2.

В табл. 1 показано результати синтезу системи згідно конфігурації №1 та №2. Порівнюючи їх, чітко видно залежність апаратних витрат від розрядності шини даних. Також суттєво зменшує апаратні витрати і складність логічних операцій. Із списку підтримуваних операцій найбільшу складність мають операції логічного зсуву. При конфігурації системи №1 є дві операції зсуву, одна – логічного, друга – арифметичного, а для конфігурації системи №2 одна операція циклічного зсуву. Це, а також зменшення кількості розрядів шини даних згідно конфігурації системи №2 і пояснює невисоке число апаратних витрат.

Таблиця. 1.

Оцінка часових та апаратних параметрів

	Необхідна кількість апаратних ресурсів для синтезу системи згідно конфігурації №1	Необхідна кількість апаратних ресурсів для синтезу системи згідно конфігурациії №2	Загальна кількість елементів на кристалі
Кількість слайсів (Slices)	287	138	42176
Тригерів (Slice Flip Flops)	1	-	84352
4 – х входові LUT	514	246	84352
Входи/Виходи IOB	76	43	768
Масимальна затримка проходженння сигналу (в нс)	12,929	11,076	-

На перший погляд можна подумати, що недоліком проектування систем на основі комбінаційної схеми є великий час проходження сигналу. Насправді, це можна пояснити тим що симуляція відбувалася для цілої системи, а не окремого вузла. Велику часову затримку створюють вхідні/вихідні буфери. Ще одним фактором що сприяє збільшенню часу проходження сигналу в

комбінаційній схемі є реалізація логічних операцій зсуву. Будь-яка операція зсуву імплементується в ярусну структуру кожен ярус якої є одинаковий на апаратному рівні.

На рис. 4 показано функціональну схему реалізації операції циклічного зсуву вправо. Розрядність шини даних складає 13 біт, кількість ярусів рівна – 4. Провівши синтез системи визначено апаратні витрати. Кількість 4-х входових LUT – 52. Відштовхуючись від того що яруси є одинакові в апаратній реалізації, можливо розробити інший варіант реалізації операції циклічного зсуву. В якому буде необхідно лише один ярус та 26 тригерів (для зберігання проміжних результатів, 13 вхідних та 13 вихідних).

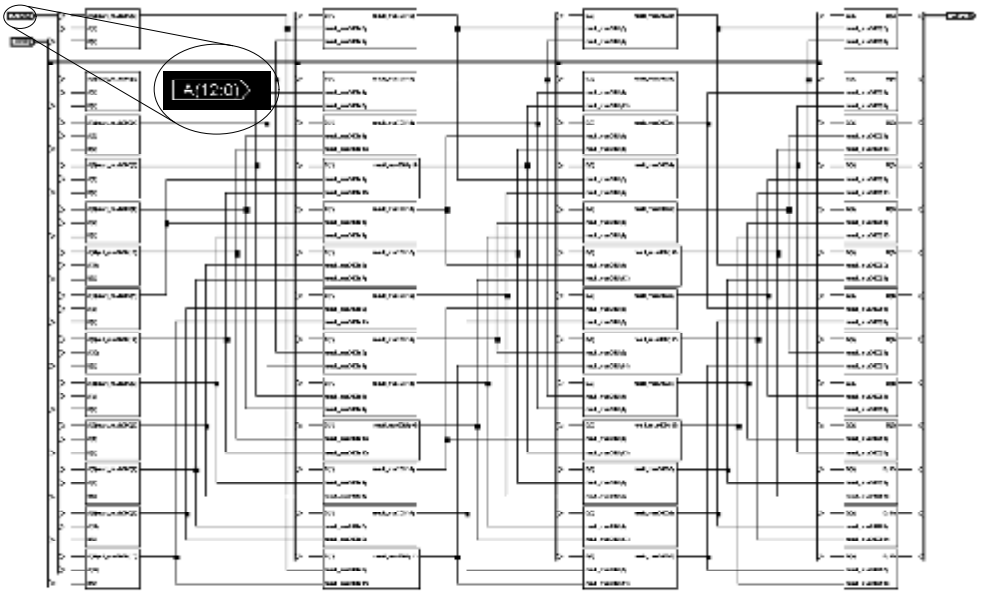

Рис. 4. Функціональна схема реалізації операції циклічного зсуву вправо.

Перевагою такого підходу є зменшення кількості апаратних ресурсів. Щодо часової затримки, вона стане меншою, але це не вплине на загальний час який необхідно затратити щоб отримати результат обчислень. Тому що для реалізації операції циклічного зсуву в одноярусній структурі необхідно 4 такти сигналу синхроімпульсу, в той час як для комбінаційної схеми лише один (зчитування з регістра результату операції). Отже, загальний час між подачею вхідних даних та отриманням результату залишається незмінним.

ВИСНОВКИ

Рекомендації щодо проектування засобів генерації та конфігурування логічних пристроїв є актуальним для проектування як арифметичних так і інших вузлів системи.

Подано функціональну схему логічного пристрою та часову діаграму роботи, результати якої підтверджують правильність роботи системи.

Використовуючи САПР Xilinx ISE 10.1 для ПЛІС Virtex 4 FX100 виконано синтез системи. Згідно результатів бачимо, що кількість апаратних витрат на реалізацію логічного пристрою як для конфігурації системи №1 та і для конфігурації системи №2 є меншою 1% від загальної кількості апаратних ресурсів даної ПЛІС, це дозволяє реалізувати масштабні проекти в яких необхідно використовувати логічні пристрої.

Проведено порівняльний аналіз проектування логічного пристрою на основі комбінаційної схеми та схеми з елементами пам'яті. Для першого, перевагою є одержання результату за один машинний такт, недоліком – невисока частота роботи системи, для другого, перевагою – мінімальні затрати апаратних ресурсів, недоліком – одержання результату за декілька машинних тактів кількість яких зростає із збільшенням розрядності шини даних.

1. Палагін А.В., Опанасенко В.Н., Сахарин В.Г. Системы верификации на основе реконфигурируемых устройств // Математичні машини і системи. - 2004. - №2. - С.100-
2. Мельник А.О. Архітектура комп'ютера: Наукове видання. - Луцьк: Волинська обласна друкарня, 2008. - 470с.
3. Xilinx Synthesis Technology (XST). User Guide / Available at http://www.xilinx.com.
4. http://www.electronics-tutorials.ws/combination/comb_1.html. 5. JASON CONG, YUZHENG DING Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays //ACM Transactions on Design Automation of Electronic Systems, Vol. 1, No. 2,. - 1996.
5. M. AYALA-RINC ´ ON, C. H. LLANOS, R. P. JACOBI,R. W. HARTENSTEIN Prototyping Time- and Space-Efficient Computations of Algebraic Operations over Dynamically Reconfigurable Systems Modeled by Rewriting-Logic //ACM Transactions on Design Automation of Electronic Systems. - 2006. - Vol. 11, No. 2. - C.251-281.