УДК 004.023 DOI: 10.18372/2073-4751.70.16842

Воронцов В.І.,
Лукашенко В.В., к.т.н.,
orcid.org/0000-0002-8898-2269

МОДИФІКАЦІЯ СУЧАСНОГО RISC ПРОЦЕСОРА ШЛЯХОМ РЕАЛІЗАЦІЇ СПЕЦІАЛІЗОВАНИХ ІНСТРУКЦІЙ

Національний авіаційний університет

vitalik1998v@gmail.com

Вступ

Існують багато алгоритмів, які використовують одні й ті самі функції. Прикладом таких функцій є тригонометричні. Тригонометричні функції застосовуються в багатьох алгоритмах цифрової обробки сигналів, наприклад перетворення Хартлі, перетворення Фур'є та у комп'ютерній графіці. Виконання даних операцій тільки при програмній реалізації відносно повільне.

Аналіз сучасних публікацій, присвячених розв'язанню задач підвищення швидкодії процесорних інструкцій показав, що увага авторів зосереджена переважно на способах підвищення шкидкодії існуючих інструкцій теоретично, проте на практиці розглядають доволі нечасто . Якщо написати процесорні інструкції що будуть сприйматися процесором як власні, тобто будуть в конвеєрі процесора, то швидкодія даних операцій зросте.

Тому було прийнято рішення розробити апаратну реалізацію обрахування функції sin(x) з використанням на основі сучасного комерційного процесору MIPSfpga та протестувати на FPGA платі.

Це необхідно для задач, де обмежена потужність процесора, наприклад вбудовані системи. Це рішення не універсальне, а спеціалізоване. На основі даної роботи можна вбудовувати нові інструкції. За аналогічною схемою до реалізації даних інструкцій можна працювати у напрямку реалізації односторонніх функцій, які необхідні для передачі інформації по відкритому каналу. Об'єктом досліджень являється процесорне ядро MIPSfpga, що є версією комерційного ядра microAptiv UP на архітектурі MIPS32.

Мета

Метою статті є підвищення швидкодії виконання процесорних інструкцій на прикладі функції sin(x) з використанням формату чисел float32 у MIPS-сумісному процесорі.

Основна частина

Для розуміння того, що розроблюється в даній роботі, необхідно ознайомитись з процесом розробки сучасних мікросхем.

Через складну природу сучасних мікросхем неможливо створити щось з нуля, і в багатьох випадках багато компонентів будуть використані повторно.

Наприклад, компанія А потребує певного модуля для взаємодії з іншими модулями в автомобілі. Вони можуть або придбати конструкцію даного модуля в іншої компанії, щоб заощадити час і зусилля, або витратити ресурси на її створення.

Непрактично спроектувати таку систему з основних конструкційних блоків, таких як тригери та CMOS-транзистори.

Замість цього розробляється поведінковий опис для аналізу дизайну з точки зору функціональності, продуктивності та інших проблем високого рівня за допомогою мови опису обладнання, наприклад Verilog або VHDL.

Зазвичай це робить цифровий дизайнер і схожий на програміста високого рівня, який володіє навичками цифрової електроніки. Коли проект RTL буде готовий, його потрібно перевірити на функціональну правильність.

Наприклад, очікується, що процесор DSP видаватиме транзакції шини з отриманням інструкцій з пам'яті і знає, що це

відбудеться, як очікувалося. На цьому етапі потрібна функціональна перевірка, яка виконується за допомогою симуляторів EDA, які можуть моделювати проект і застосовувати до нього різні стимули. Це робота інженера з верифікації.

Щоб заощадити час і досягти функціонального завершення, як проекти, так і групи верифікації працюють паралельно, де дизайнери випускають версію RTL. Команда верифікації розробляє середовище тестового стенду та тестові випадки, щоб перевірити функціональність цієї версії RTL.

Якщо будь-який з цих тестів не вдається, це може свідчити про проблему з дизайном, і на цьому елементі дизайну буде виникнути помилка. Цю помилку потрібно буде виправити в наступній версії випуску RTL від команди дизайнерів. Цей процес триває до тих пір, поки не з'явиться хороший рівень впевненості у функціональній коректності конструкції.

На наступному етапі цей дизайн перетворюють на апаратну схему з реальними елементами, такими як комбінаційні схеми та тригери. Цей етап називається синтезом.

Інструменти логічного синтезу дозволяють конвертувати опис RTL у HDL у список зв'язків схем на рівні логічних вентилів. Цей список мереж є описом схеми в термінах вентилів і з'єднань між ними.

Інструменти логічного синтезу забезпечують відповідність списку зв'язків специфікаціям часу, площі та потужності. Як правило, вони мають доступ до різних технологічних вузлів процесів і бібліотек цифрових елементів і можуть робити інтелектуальні розрахунки, щоб відповідати всім цим різним критеріям.

Ці бібліотеки отримані від напівпровідникових фабрик, які надають характеристики даних для різних компонентів, таких як час наростання або спаду для тригерів, час введення-виведення для комбінаційних вентилів тощо.

Список зв'язків схем на рівні шлюза перевіряється на логічну еквівалентність RTL. Іноді перевірка на рівні шлюза виконується, коли перевірка певних елементів виконується ще раз, різниця в тому, що цього разу це відбувається на рівні воріт і нижчому рівні абстракції. Час симуляції, як правило, повільніший через величезну кількість елементів, що беруть участь у проектуванні, і інформацію про затримку.

Потім список зв'язків схем вводиться до фізичного процесу проектування, де виконуються розміщення і трасування за допомогою інструментів EDA. Це вибере та помістить стандартні комірки в рядки, визначить введення та виведення, створить різні металеві шари та розмістить буфери.

Після завершення цього процесу створюється макет, який зазвичай відправляється на виготовлення. Цим етапом зазвичай займається команда фізичного проектування, яка добре знайома з технологічним вузлом і деталями фізичної реалізації.

Зразок мікросхеми буде виготовлено тією ж компанією з виробництва напівпровідників або надіслано сторонній стороні, наприклад всім відомий завод TSMC. Цей зразок тепер проходить процес перевірки після кремнію, коли інша команда інженерів запускає різні шаблони тестування. Це складніше налагодити під час перевірки після реалізації в кремнії, ніж перед цим шляхом, просто тому, що рівень видимості внутрішніх вузлів чіпа різко знижується. Через те, що мільйон тактових циклів завершився б за секунду, відстеження до точного часу помилки займе багато часу. Якщо на цьому етапі будуть виявлені якісь реальні проблеми або помилки в дизайні, це потрібно буде виправити в RTL, повторно перевірити, а також виконати всі наступні кроки.

Незважаючи на те, що в потоці проектування є кілька етапів, велика частина проектної діяльності зазвичай зосереджена на оптимізації та перевірці опису RTL схеми.

Розробка, проведена в даній робота зосереджена на написанні на перевірці RTL схеми. Зміни рівня RTL вводяться у

ядро MIPSfpga, що є версією комерційного ядра microAptiv UP. Процесори microAptiv знаходять широке застосування в комерційних виробах, включаючи пристрої промислової і офісної автоматизації, автомобільної та споживчої електроніки, засоби бездротового зв'язку. Ядро MIPSfpga було спроектовано з використанням однієї з мов опису апаратури Verilog. Воно являє собою конфігуроване (програмоване) ядро (soft core), оскільки воно описано на мові Verilog, а не фізично реалізовано у вигляді мікросхеми. [1].

Для розрахунку функції sin(x) необхідно обрати метод. У розрахунку функції sin(x) є деяка особливість. Аргумент, як правило, визначається лише для першого квадранта, тобто 0≤ x ≤ π/2, а інші значення квадрантів обчислюються через наступні формули:

$$sin(x) = -sin(-x) \quad (1)$$

$$sin(x) = sin(\pi/2 - x) \quad (2)$$

Рис. 1 показує точне значення та наближення для 16-розрядного квантування:

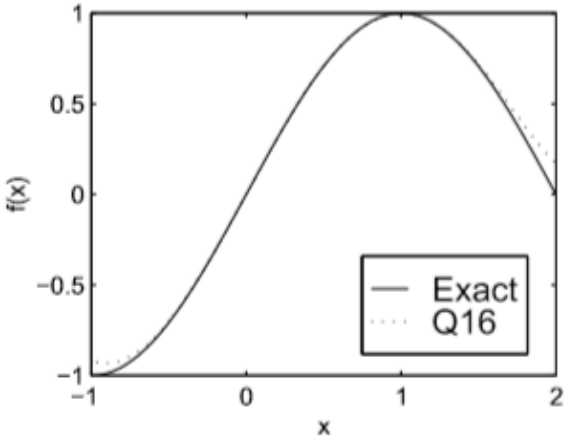

Рис.1. точне значення та наближення для 16-розрядного квантування

На рис. 2 відображається похибка, тобто різниця між точними значеннями функції та наближенням:

Є декілька способів обчислення синуса. Наприклад, поліном Чебишева або поліном Тейлора. Проблема поліному Чебишева полягає в тому, що він визначений лише для діапазону x ∈ [−1,1]. [2].

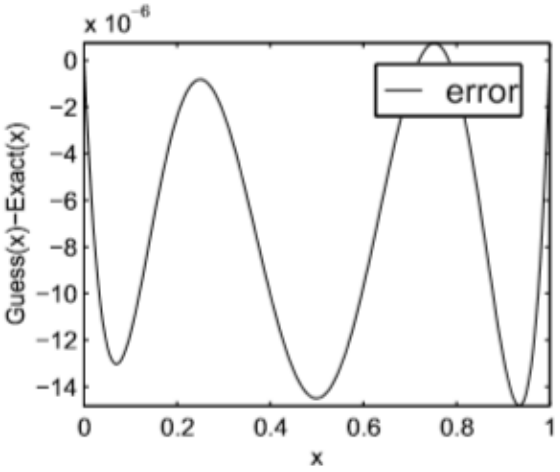

Рис. 2. Різниця між точними значеннями функції та наближенням

В той же час, у полінома Тейлора такої проблеми немає. Тому для обрахування синуса було обрано даний метод:

$$sin(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1} \quad (3)$$

При цьому використали 4 доданки:

$$sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \quad (4)$$

На рис. 3 зображений Графік наближення поліному:

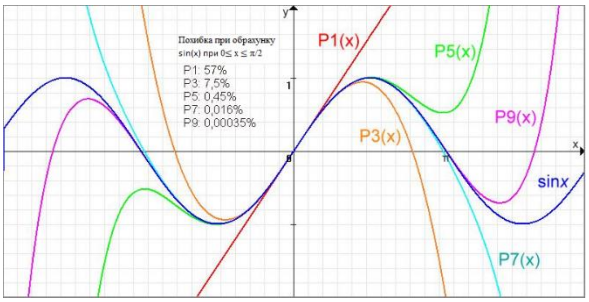

Рис. 3. Графік наближення поліному

При необхідності завжди можна збільшити кількість доданків у результуючому коді.

Для інтеграції sin(x) в MIPS-сумісний процесор був написаний модуль обрахування синуса, алгоритм якого у зрозумілому вигляді виглядає наступним чином:

```
y = x1 * 2 / 3.14159265;
y2 = y * y;
y3 = y * y2;
y5 = y3 * y2;
y7 = y5 * y2;
sum = 1.57 * y - 0.645 * y3 +
      0.08 * y5 - 0.00472 * y7;
```

Даний модуль використовує float32 додавання та множення. Сам модуль продемонстрований у додатку А.

Після того як отримано необхідний вихідний Verilog-код, необхідно перейти до етапу інтеграції в процесор.

Система MIPS FPGA має спеціальний модуль, що називається UDI та розшифровується як інструкції, визначені користувачем. Це ефективний механізм для розширення функціоналу в MIPS-процесорі. Даний модуль підтримує до 7 додаткових інструкцій. В даній інтеграції використовується лише 5.

Створено новий модуль, що називається m14k_udi_seleqz. Використовується формат інструкції SPECIAL2, що має 3 аргументи та спеціальне поле для визначення сценарію використання інструкції, тобто кожну інструкцію можна поділити на ще більше інструкцій за допомогою коду в спеціальному полі. Оскільки використано всього 4 інструкції, потреби в цьому не було, тому постійно в спеціальне поле передається значення 0. Формат інструкції представлено на рис. 4.

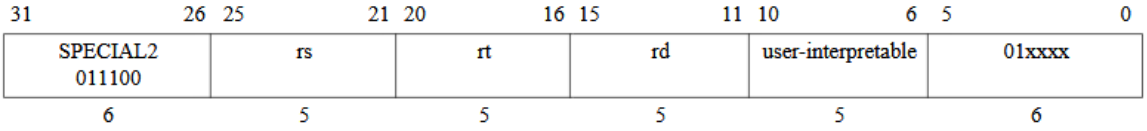

Рис. 4. Формат інструкції UDI

Для тестуванні на платі було застосоване середовище розробки на базі Visual Studio Code. Було проведено ряд тестів. Для прикладу одного з екземплярів тестової одиниці, візьмемо наступні дані: аргументом є $3f255150_{16}$ у форматі float32, що розшифровується як 0.645772 радіан (37 градусів).

На виході маємо число 1058672768_{10}, що дорівнює $3f1a1080_{16}$. У результаті отримано число sin(0.645772) = 0.601814270.

В результаті тестування отримали, що результуючий модуль має похибку відносно точного значення функції sin(x) приблизно 0.1%.

Висновки

Вперше запропонована інструкція для обрахування функції sin(x) за допомогою полінома Тейлора, яка була впроваджена у процесорне ядро MIPSfpga, та протестована у пакеті ModelSim та на FPGA платі Altera DE2-115. Результати роботи можна використати для вивчення роботи процесора, ознайомлення з інструкціями користувача, покращення поточної реалізації, для модернізації існуючих інструкцій та реалізації нових процесорних інструкцій, використовуючи розроблені модулі.

Література

1. *Harris D., Harris S.*. Digital Design and Computer Architecture, 2nd Edition. – 2012. – 720 p.

2. *Uwe Meyer-Baese*. Digital Signal Processing with Field Programmable Gate Arrays, Fourth Edition: навчальний посібник. – Springer, 2005. – 930 c.

Воронцов В.І., Лукашенко В.В.

МОДИФІКАЦІЯ СУЧАСНОГО RISC ПРОЦЕСОРА ШЛЯХОМ РЕАЛІЗАЦІЇ СПЕЦІАЛІЗОВАНИХ ІНСТРУКЦІЙ

Існують багато алгоритмів, які використовують одні й ті самі функції. Прикладом таких функцій є тригонометричні. Тригонометричні функції застосовуються в багатьох алгоритмах цифрової обробки сигналів, наприклад перетворення Хартлі, перетворення Фур'є та у комп'ютерній графіці.

Виконання даних операцій тільки при програмній реалізації відносно повільне. Якщо написати процесорні інструкції, що будуть сприйматися процесором як власні, тобто будуть в конвеєрі процесора, то швидкодія даних операцій зросте.

Було розроблено апаратну реалізацію обрахування функції sin(x) на основі сучасного комерційного процесору MIPSfpga та протестованого на платі DE2-115.Це необхідно для задач, де обмежена потужність процесора, наприклад вбудовані системи. Це рішення не універсальне, а спеціалізоване.

Вперше запропонована інструкція для обрахування функції sin(x) за допомогою полінома Тейлора, яка була впроваджена у процесорне ядро MIPSfpga, та протестована у пакеті ModelSim та на FPGA платі Altera DE2-115. Результати роботи можна використати для вивчення роботи процесора, ознайомлення з інструкціями користувача, покращення поточної реалізації, для модернізації існуючих інструкцій та реалізації нових процесорних інструкцій, використовуючи роз-роблені модулі.

Ключові слова: *FPGA, інструкції визначені користувачем, асемблер, мови опису апаратури, процесор.*

Vorontsov V.I., Lukashenko V.V.

MODIFICATION OF MODERN RISC PROCESSOR BY IMPLEMENTATION OF SPECIALIZED INSTRUCTIONS

There are many algorithms that are using the same functions. Examples of such functions are trigonometric ones. Trigonometric functions are used in many digital signal processing algorithms, such as the Hartley transform, the Fourier transform, and in computer graphics.

The execution of these operations only with software implementation is relatively slow. If we can write processor instructions that will be perceived by the processor as their own, that will be in the processor pipeline, then the speed of these operations will increase.

A hardware implementation of the sin(x) function calculation was developed based on a modern commercial MIPSfpga processor and tested on the DE2-115 board. This is necessary for tasks where the processor power is limited, such as embedded systems. This solution is not universal, but specialized.

For the first time, an instruction for calculating the sin(x) function using the Taylor polynomial was proposed, which was implemented in the MIPSfpga processor core, and tested in the ModelSim package and on the Altera DE2-115 FPGA board. The results of the work can be used to study the operation of the processor, familiarize yourself with the user instructions, improve the exact implementation, to modernize the existing instructions and implement new processor instructions using the developed modules.

Keywords*: FPGA, user defined instruction, assembler, hardware description language, processor.*