Побудова та аналіз завадостійких процесорних архітектур

Ю.О. Гунченко
кафедра математичного забезпечення комп'ютерних систем
Одеський національний університет імені І.І.Мечникова
Одеса, Україна

П.С. Ємельянов
кафедра математичного забезпечення комп'ютерних систем
Одеський національний університет імені І.І.Мечникова
Одеса, Україна

Е.Ю. Морозова
кафедра математичного забезпечення комп'ютерних систем
Одеський національний університет імені І.І.Мечникова
Одеса, Україна

Е.А. Любкевич
кафедра математичного забезпечення комп'ютерних систем
Одеський національний університет імені І.І.Мечникова
Одеса, Україна

Construction and analysis of jam protected CPU design

Y. Gunchenko
department of the mathematical providing of the computer systems
Odessa national university of the name I.I. Mechnikov
Odessa, Ukraine

P. Emelianov
department of the mathematical providing of the computer systems
Odessa national university of the name I.I. Mechnikov
Odessa, Ukraine

E. Morozova
department of the mathematical providing of the computer systems
Odessa national university of the name I.I. Mechnikov
Odessa, Ukraine

E. Lyubkevich
department of the mathematical providing of the computer systems
Odessa national university of the name I.I. Mechnikov
Odessa,

***Анотація* — В роботі проаналізовано принципи побудови завадостійких видів архітектури процесорів для сучасних обчислювальних систем. Описано функціонування і особливості модернізації для трьох видів архітектури, показано їх переваги та недоліки, наведено залежності швидкості роботи (кількості інструкцій, що виконуються) від ймовірності помилки обчислювань.**

***Abstract* — In this paper the principles of jam protected CPU design for modern computing systems are analyzed. Functioning and features of modernization for three CPU designs are described, their advantages and disadvantages are shown, dependences of work's speed (number of executed instructions) from probability of an error of calculations are given.**

Ключові слова — процесор, завадостійка архітектура, сигнатура, обчислювальна система.

Keywords — processor, CPU design, signature, computing system.

I. Вступ

У сучасних обчислювальних системах спеціального призначення часто стоїть проблема надійності. При цьому вирішуються три типи завдань:

- підвищення надійності зберігання даних;
- підвищення надійності передачі даних;
- підвищення надійності обчислень (достовірності отриманих даних).

Завдання першого та другого типів спрямовані на виявлення та усунення помилок, що виникають при

передачі або зберіганні даних, і основи їх вирішення мають математичний або фізичний характер. Існує безліч різних способів підвищення надійності зберігання та передачі даних: використання контролю (біта) парності, кодів Хеммінга, Хаффмана і інших, більш складних кодів.

Рішення ж завдань третього типу пов'язано зі змінами архітектури обчислювальних пристроїв і самої системи в цілому.

Необхідність підвищення надійності обчислень найбільш сильно відчувається при організації розподілених обчислень, а також побудові розподілених систем управління різноманітного призначення. Проблема підвищення надійності обчислень на даний момент є актуальною, а її рішення - користуються попитом. Підвищення надійності обчислень за допомогою зміни архітектури може бути як непрямим, так і безпосередньо впливати на надійність обчислень.

II. Побудова Та Аналіз Завадостійких Процесорнихархітектур

Архітектура 1. На рис. 1 представлена структурна схема завадостійкого процесора. Блок виконання команд 101 з'єднаний з лічильником команд 102, файл-регістром 103, буфером запису 104, блоком генерації сигнатури 105 і блоком контролю помилок 106. Лічильник команд 102, файл-регістр 103 і регістр сигнатури 107 з'єднані з відповідними контрольними регістрами 108, 109, 110.

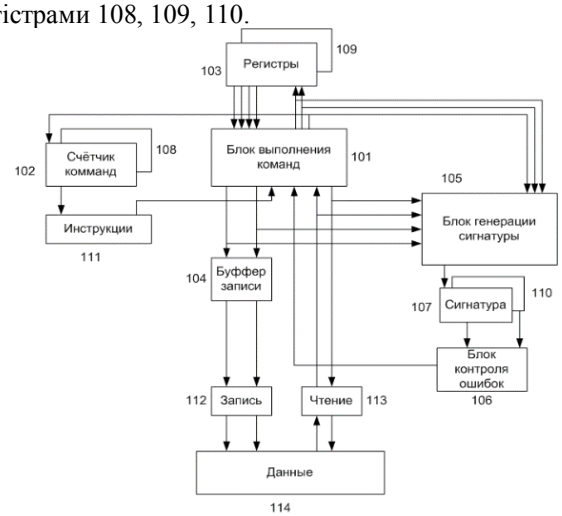

Рис. 1. Структура процесора

Файл-регістр 103 має контрольний регістр 109, який може зберігати поточний стан файл-регістра (створення контрольної точки) і може відновлювати значення файл-регістра з контрольної точки. Лічильник команд 102 має контрольний регістр 108, здатний зберігати поточний значення лічильника команд і відновлювати його значення. Блок виконання команд 101 виконує інструкції, що містяться в пам'яті 111, на які вказує лічильник команд 102. Інструкції зчитуються і змушують блок виконання команд 101 оперувати вмістом файл-регістра 103. Блок виконання команд 101 також може зберігати дані 112 і завантажувати 113 з пам'яті 114. Буфер запису 104 містить дані, які повинні бути записані в пам'ять. Ця операція виконує дві функції. По-перше, буфер запису 104 звільняє процесор від зупинок під час операцій записи. По-друге, буфер запису 104 містить дані, які можуть бути використані або відкинуті в разі виникнення помилки. Зауважимо, що буфер запису 104 не може бути джерелом даних, якщо адреса даних при завантаженні збігається з адресою даних, записаних у буфері запису 104. Це обмеження необхідне для досягнення відмовостійкості даної архітектури. Блок генерації сигнатур 105 приймає всі результати обчислені блоком виконання команд 101. До них також відносяться всі завантажені дані. Спосіб обчислення сигнатури в даній роботі не розглядається. Блок генерації сигнатур 105 зберігає обчислену сигнатуру до відповідного регістру 107. Регістр сигнатури 107 має свій контрольний регістр 110, в який записується поточне значення сигнатури, при цьому контрольна копія сигнатури 110 ніколи не записується назад в регістр сигнатур 107. Блок контролю помилок 106 контролює запис в контрольні регістри 108, 109, 110, контролює очищення буфера запису 104 і порівнює поточну сигнатуру 107 з попередньої 110.

У стані "0" процесор виконує інструкції для формування сигнатури, яка буде використовуватися в наступному стані для виявлення помилки.

У стані "1" заново виконуються всі команди для визначення наявності помилки. Якщо сигнатури збігаються, то помилки не було. Процесор переходить в стан "0". Якщо поточна сигнатура не збігається з попередньою, значить виявлена помилка і процесор переходить в стан "2".

Стан "2" починається з очищення буфера запису. Це робиться у зв'язку з тим, що була виявлена помилка і дані, що зберігаються в ньому, можуть бути невірними. Після всіх дій в стані "2", процесор переходить в стан "0".

Архітектура 2. Внесемо невеликі зміни в вихідну архітектуру і алгоритм функціонування, а саме змінимо алгоритм в стані "2".

Відмінність від початкової архітектури тільки в одному з блоків і в кінцевому переході в стан "1". Таким чином, якщо перехід в стан "2" не був здійснений (не було помилки), то відмінностей між даною архітектури та початкової не буде. Якщо ж перехід в стан "2" був здійснений (була помилка), то очищається буфера запису. Стан "1" буде починатися в тих самих початкових умовах, що і в попередній раз за винятком того, що в контрольному регістрі сигнатури 110 буде зберігається нове значення сигнатури. Внесена зміна призвела до того, що блок контролю помилок вважатиме результат вірним, якщо його сигнатура збігається з сигнатурою попереднього результату.

Архітектура 3. Архітектуру процесора можна додатково модифікувати, шляхом додавання контрольного регістра сигнатури 2 і мажоритарного елементу та відповідною зміною станів "1" і "2".

Після підрахунку сигнатури виконується її порівняння з сигнатурами, що зберігаються в контрольних регістрах за допомогою мажоритарного елементу. У контрольному регістрі на момент порівняння зберігається сигнатура попереднього

результату. Контрольний регістр на момент першого порівняння проініціалізовано, а при наступних порівняннях буде містити сигнатуру, отриману на попередньому кроці.

Таким чином, блок контролю помилок вважатиме результат правильним, якщо його сигнатура збігається з однією з двох попередніх сигнатур.

III. Порівняння Архітектур

Визначимо ймовірності отримання помилкового результату (помилковою сигнатури) для трьох розглянутих архітектур процесорів.

Приймемо ймовірність отримання помилкової сигнатури на кожному циклі (кроці) рівноймовірної та рівній p, тоді ймовірність отримання вірної сигнатури q = 1 - p. Виконання команди менш ніж за два цикла неможливо ні в одній з трьох розглянутих архітектур. Позначимо через Pi (n) ймовірність виконання команди на n-му кроці в i-й архітектурі. Виконання команди менш ніж за 2 кроки в розглянутих архітектурах неможливо і тому Pi(n)=0, при n <2 для будь-якого i. Також відзначимо, що P (2) = q2 для будь-якої архітектури, тому що відмінності в них спостерігаються тільки при виявленні помилки, а прийняття результату на другому кроці свідчить про її відсутність.

В архітектурі 1.

$$P_1(n=2k)=q^2(1-qq)^{k-1} \qquad (1)$$

В архітектурі 2.

$$P_2(n)=p(p-1)^2(1-F(n-3)) \qquad (2)$$

де $F(n)=\sum_{i=1}^{n}P(i)$ – функція розподілу, що дорівнює ймовірності отримання правильного результату за n-менш циклів.

В архітектурі 3 з використанням мажоритарного елементу.

$$P_3(n)=(1-F(n-4))(pq)^2+ \\ +(1-F(n-5))p(pq)^2 \qquad (3)$$

або

$$P_3(n)=(1-F(n-4)+pF(n-5))(pq)^2 \qquad (4)$$

де $F(n)=\sum_{i=1}^{n}P(i)-$ функція розподілу, що дорівнює ймовірності отримання правильного результату за n-менш циклів.

Середня кількість циклів (фактично інструкцій), за які буде виконана поодинока інструкція для будь-якої архітектури визначається математичним очікуванням функції Pi (n) відповідної архітектури, і безпосередньо залежить від імовірності помилки p:

$$N_{cp}=\sum_{i=1}^{\infty}iP(i)\,.$$

На Рис. 2, 3 наведені залежності середнього числа виконуваних циклів від ймовірності помилки p. Вихідній архітектурі відповідають графіки 1, модифікованої - 2, архітектурі з використанням мажоритарного елементу - 3.

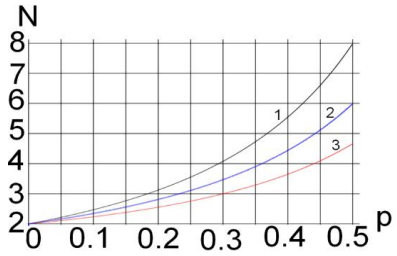

Рис. 2. Среднее число циклов при p<0,5

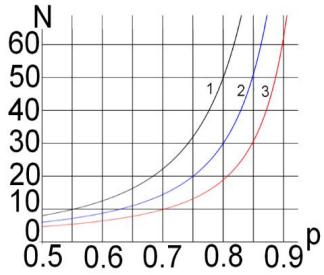

Рис. 3. Среднее число циклов при p>0,5

Як видно з Рис. 2, 3, при будь-якій ймовірності помилки p для архітектури 3 з мажоритарним елементом середня кількість циклів (виконання інструкцій) для отримання результату мінімальна, а для вихідної архітектури 1 - максимальна.

Висновки

У роботі проаналізовано принципи побудови завадостійких архітектур процесорів для сучасних обчислювальних систем. Описано функціонування і особливості модернізації для трьох архітектур, показані їх переваги та недоліки, наведені залежності швидкості роботи (кількості виконуваних інструкцій) від ймовірності помилки обчислень

Література References

[1] Затуливетер Ю.С., Фищенко Е.А, Ходаковский И.А. Разработка и исследование методов повышения надёжности распределенных вычислений. Журнал "Надёжность" 2009г. Вып.1, с. 42-49.

[2] Матушевский В.В., Гунченко Ю.А. Вычислительные системы (учебное пособие). – Одесса: ВМВ, 2011. 204 с.

[3] Donald E. Steiss, Single event upset tolerant microprocessor architecture – [Елктронний ресурс] – Режим доступу: http://patft.uspto.gov/netacgi/nphParser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6,571,363.PN.&OS=PN/6,571,363&RS=PN/6,571,363.

[4] Гнеденко Б.В. Курс теории вероятности: Издание шестое, Москва "Наука" 448с.

[5] Иванов М.А., Чугунков И.В., Теория, применение и оценка качества генераторов, М.: КУДИЦ-ОБРАЗ, 2003.

[6] Пат 67752 Україна, МПК(2006.01) G06F 12/08. Пристрій підвищення завадостійкості систем з програмним управлінням / Гунченко Ю.О., Мартинюк С.М., Лєнков С.В., Омельченко О.С., Купрацевич А.В.; власник Одеський національний університет ім. І.І. Мечникова. - № u201107423, заявл. 14.06.2011, опубл. 12.03.2012, Бюл. № 5.

[7] Омельченко А.С, Мигов И.В., Гунченко Ю.А, Архитектура процессора для отказоустойчивых вычислительных систем. // Восьма Регіональна конференція студентів і молодих науковців. 2011р, с.17-18.

[8] Пат 76984 Україна. МПК (2006.01) G06F 11/27. Відмовостійкий процесорний пристрій з підвищеною швидкодією / Гунченко Ю.О., Лєнков С.В., Кобозєва А.А., Мартинюк С.М., Борисенко І.І.; власник Одеський національний політехнічний університет. - №u2012 07958, заявл. 27.06.2012, опубл. 25.01.2013, Бюл. №2.

[9] Гунченко Ю.О., Омельченко А.С., Пенко О.А. Завадостійкий процесорний пристрій з підвищеною швидкодією. // Тези доповідей Третьої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації» 2011р с. 42