**2. Масштабирование чисел в RNS с использованием функции ядра Акушского**

**2.1 Функция ядра Акушского**

Функция ядра Акушского определена для целого числа, , как:

|  |  |
| --- | --- |
|  | (1) |

где обозначает иобозначает -ый вес. Подставив в (1) получим:

|  |  |
| --- | --- |
|  | (2) |

|  |  |
| --- | --- |
|  | (3) |

Это означает, что некоторые значения должны быть отрицательными, чтобы получить малые значения . Подставив (3) в (1), получим:

|  |  |
| --- | --- |
|  | (4) |

Уравнение (4) указывает, что график относительно должен быть прямой линией с наклоном с некоторой «пушистостью». Величина пушистости определяется величиной весов, в свою очередь связанной с конкретным значением для заданного набора модулей RNS.

Веса, , определяются путем перестановки (3) и сокращения обеих сторон :

|  |  |
| --- | --- |
|  | (5) |

Учитывая, что для любого , константы этой функции могут быть определены из соотношения

|  |  |
| --- | --- |
|  | (6) |

таким образом, веса могут быть определены после выбора но с условием выполнения (2), что подразумевает, что некоторые веса должны быть отрицательными, чтобы обеспечить . Обратите внимание, что если является кратным , то соответствующий вес . Наконец, из [10] можно узнать, что диапазон функции ядра Акушского, задан как:

|  |  |
| --- | --- |
|  | (7) |

Китайская теорема об остатках для преобразования чисел обратно в позиционное (то есть десятичное или двоичное) представление может быть выражена следующим образом:

|  |  |
| --- | --- |
|  | (8) |

где известна как функция ранга, а ортогональный базис RNS:

|  |  |
| --- | --- |
|  | (8) |

Подставляя (8) в (4), получаем:

|  |  |
| --- | --- |
|  | (9) |

Упрощая и переписывая (10), получаем:

|  |  |
| --- | --- |
|  | (11) |

Подставим в (4):

|  |  |
| --- | --- |
|  | (12) |

Отсюда:

|  |  |
| --- | --- |
|  | (13) |

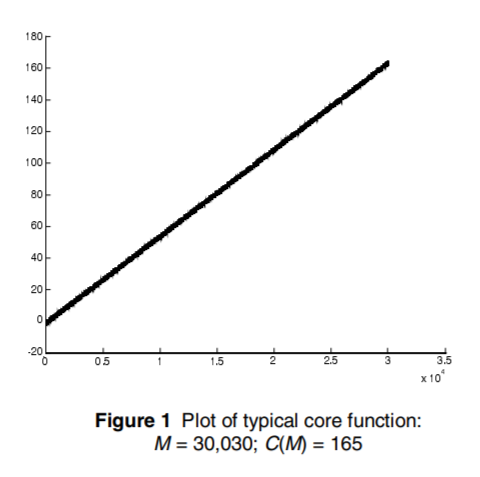
Данная форма известна как Китайская теорема об остатках для функции ядра Акушского. Однако, из-за того, что вычисление невозможно независимо, предпочтительная форма Китайской теоремы об остатках функции ядра Акушского это:

|  |  |
| --- | --- |
|  | (14) |

**Пример 1.** Рассмотрим RNS с набором модуля, , что даёт , и . Затем выберите . Затем, из (6), веса находятся следующим образом:

Чтобы минимизировать «пушистость» в функции ядра Акушского, следует выбирать веса с малыми значениями. В этом примере выбран набор весов , его легитимность можно проверить по уравнению (2):

Эта функция ядра Акушского изображена на рисунке 1. Имеется альтернативный набор весов , который обладает полезным свойством если . Легитимность этого набора весов также можно проверить следующим образом:



**Рисунок 1** График типичной функции ядра Акушского

**2.2 Метод масштабирования RNS**

Эта статья предлагает метод масштабирования RNS, который состоит в извлечении ядра числа внутри RNS. Из уравнения (4):

|  |  |
| --- | --- |
|  | (4) |

Таким образом, если может быть вычислено внутри RNS, то получается приближенная масштабированная версия числа . Это можно достичь, разделив набор модулей на два поднабора, и , так что и . Затем возможно выполнить масштабирование либо , либо (то есть извлечь с или ), следуя следующей процедуре.

Сначала установим . Затем, из (14), получим:

|  |  |
| --- | --- |
|  | (15) |

для поднабора модулей, составляющих . Но, является делителем , так что:

|  |  |
| --- | --- |
|  | (16) |

Однако для поднабора модулей, составляющих - то есть модули, , которые не делятся на - то же упрощение невозможно:

|  |  |
| --- | --- |
|  | (17) |

Таким образом, уравнение (15) может быть вычислено в поднаборе модулей, , но (16) - нет. Аналогично, если :

|  |  |
| --- | --- |
|  | (18) |

что может быть вычислено в поднаборе модулей, . Таким образом, может быть вычислено внутри поднабора модулей,, но не внутри поднабора, ; также, может быть вычислено внутри поднабора модулей, , но не внутри поднабора, . Однако, если разница между ядрами может быть вычислена, по модулю поднабора может быть расширено до по модулю поднабора . Другими словами, добавляя к (или вычитая его из) значения одного из поднаборов масштабированных остатков, либо , либо становится доступным для всех остатков, и получается масштабированное значение внутри RNS. Простое выражение для получается из (13) следующим образом:

|  |  |
| --- | --- |
|  | (19) |

которое можно упростить до:

|  |  |
| --- | --- |
|  | (20) |

где , и . Однако, учитывая сложность определения значенияпо остаткам, более полезной формой уравнения (20) является:

|  |  |
| --- | --- |
|  | (21) |

Это выражение будет наиболее удобно оценивать, если имеет схожую длину слова с модулями (или небольшой степень двойки).

**2.3 Отработанный пример предложенного алгоритма масштабирования RNS**

Предположим, у RNS есть набор модулей:,: тогда . Набор модулей разделен на две группы,и , в результате . Масштабирование числа остатков на 2737 или 2717 является практичным, потому что одинаковые значения по модулю.

Значения и равны и соответственно. Набор весов для равен и для , . Два набора следуют из (12): ; ; наконец

Число должно быть примерно масштабировано на 2717 чтобы получить . То есть, мы будем вычислять полностью в рамках RNS. Число представлено в виде в этом наборе модулей. Сначала вычислим по модулям 7, 17 и 23, и по модулям 11, 13 и 19, используя (15) и (17):

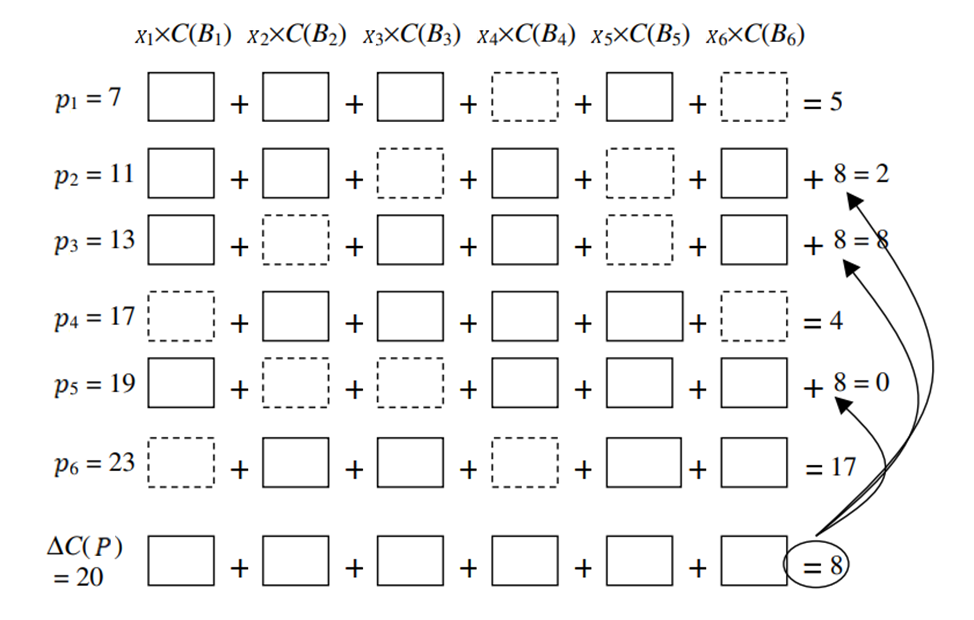
Параллельно вычислим , используя (21):

Наконец, добавим к значениям , чтобы получить оставшиеся масштабированные модули:

Таким образом, RNS-значение (или в формате RNS) после приближенного масштабирования на 2717 равно

Мы можем проверить результат, преобразовав обратно в десятичное число с использованием Китайской Теоремы об остатках:

Блок-схема этого метода расчета, с акцентом на последовательное использование арифметики с короткой длиной слова и явным указанием доступной степени параллелизма, представлена на Рисунке 2.



**Рисунок 2** Пример вычисления масштабирования RNS

Каждый блок на Рисунке 2 представляет собой таблицу поиска в ROM, в которой сохранены возможные кратные значений , где обозначает номер столбца, а q - номер строки. Обратите внимание, что некоторые ROM на Рисунке 2 могут быть удалены, поскольку соответствующие коэффициенты равны 0 - эти ROMs обозначены пунктирными линиями. Общее число операций MAC равно , что почти в два раза меньше, чем в методе Shenoy и Kumaresans.

**2.4 Анализ ошибок алгоритма масштабирования**

Ошибка,, которая возникает при использовании функции ядра Акушского для представления масштабированного числа вместо округленного деления, может быть оценена следующим образом:

|  |  |
| --- | --- |
|  | (22) |

где - целое число. Подставляя в уравнение (4), получаем:

|  |  |
| --- | --- |
|  | (23) |

Остаток от деления на равен:

|  |  |
| --- | --- |
|  | (24) |

где . В таком случае,

|  |  |
| --- | --- |
|  | (25) |

Таким образом, наибольшая положительная ошибка в возникнет, когда все остатки с модулями с положительными весами будут наибольшими:

|  |  |
| --- | --- |
|  | (26) |

где обозначает положительные веса. Аналогично, наибольшая отрицательная ошибка в возникнет, когда все остатки с модулями с отрицательными весами будут наибольшими:

|  |  |
| --- | --- |
|  | (27) |

где обозначает отрицательные веса. Таким образом, для минимизации ошибки в масштабированном числе следует выбирать функцию ядра Акушского с меньшим общим весом по модулю (т.е. меньшим значением . В приведенном примере, где набор весов для равен , а для равен , выбрана была .

**3. Неоднозначность в извлечении основной функции**

Существует ряд трудностей, связанных с "неопределенностью" функции ядра Акушского, которые, если не устранить, могут ограничить эффективность метода:

* , или могут быть отрицательными для некоторых значений .
* или могут превысить или соответственно.
* может превысить .

Такие случаи могут вызвать трудности, поскольку уравнения (14) и (21) вычисляются в конечных полях, что может привести к получению результатов за пределами допустимого диапазона, но ошибочных. Например, если , которая неявно вычисляется по mod , является отрицательной, будет неправильно возвращено значение . Аналогично, если , будет неправильно возвращено значение .

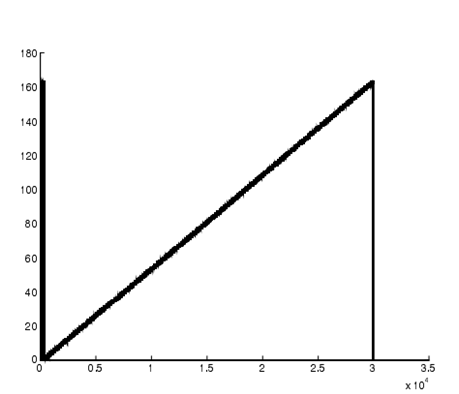
**3.1 Примеры двусмысленности**

Для иллюстрации этих проблем предположим, что система RNS имеет набор модулей, состоящий из шести наименьших простых чисел: , образующих . Набор модулей разделен на две группы, и , что дает . Масштабирование числа остатков по модулю 165 или 182 является практичным, так как имеет схожую длину слова с модулями.

Значения и равны и соответственно. Множество весов для равно , а для равно . Два набора затем следуют из (11): наконец, . Рисунок 1 (представлен ранее в разделе 2) показывает график относительно *X*, вычисленный с использованием (1): то есть, вне RNS, где не может возникнуть неоднозначность.

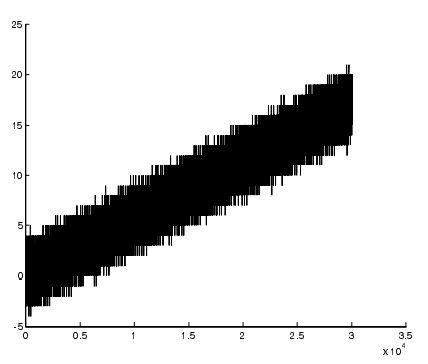
Рисунок 3 это график той же функции, но теперь вычисленного по модулю , соответствующего уравнению 14. Обратите внимание на две небольшие области неоднозначности на обоих концах оси X. Во многих приложениях RNS эти области можно избежать, выбрав набор модулей с большим динамическим диапазоном, чем у приложения.

Однако в этом использовании функции ядра Акушского неоднозначность избегается для значений (то есть, когда ), потому что ядро Акушского эффективно извлекается по модулю M, а не по модулю . Следовательно, нет неоднозначности, связанной с алиасингом, возникающим из уравнения (4). Однако алиасинг может возникнуть для значений (то есть, когда ), потому что может быть отрицательным. Два возможных решения для этого: (i) выбрать набор весов, который предотвращает отрицательное значение ; (ii) добавить небольшое смещение после методики масштабирования, эквивалентное наиболее отрицательному значению, которое может принять .

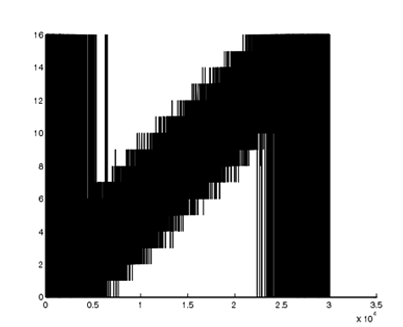


**Рисунок 3** График основной функции

Рисунок 4 представляет график для предыдущего примера.



**Рисунок 4** Участок из

Рисунок 5 также представляет собой график , но вычисленного по модулю , как в алгоритме масштабирования. Обратите внимание, как область неопределенности гораздо больше, чем на графике функции ядра Акушского. Это происходит потому, что "неопределенность" графика, приближенно выраженная суммой величин весов [10], составляет значительную часть модуля. То есть . Предложенный алгоритм масштабирования направлен на воспроизведение функции ядра Акушского, изображенной на рисунке 1, внутри RNS. Однако неопределенность, иллюстрирующаяся на рисунке 5, мешает этому происходить в большей части диапазона *X*. Таким образом, основное препятствие для масштабирования RNS с использованием функции ядра Акушского заключается в неопределенности при вычислении 

**Рисунок 5** Участок из

**3.2 Устранение двусмысленности при вычислении различия функций ядра Акушского**

В [10] были предложены два метода преодоления неопределенности в вычислениях функции ядра Акушского, оба из которых основывались на сохранении бита четности на всех этапах системы обработки RNS. В данной статье предлагается похожая идея для преодоления неопределенности в вычислении путем использования бита четности для вычисления в пределах конечного поля

Ранее было представлено уравнение (21), которое используется для вычисления :

|  |  |
| --- | --- |
|  | (21) |

Теперь возникает неопределенность в этом уравнении, потому что диапазон превышает . Ранее было показано простое выражение для диапазона функции ядра Акушского

|  |  |
| --- | --- |
|  | (7) |

Таким образом, выражение для максимального диапазона можно записать как:

|  |  |
| --- | --- |
|  | (28) |

Это следует из того, что два набора весов ортогональны: для каждого модуля в RNS один из весов в двух функциях ядра Акушского должен быть равен нулю. Это означает, что должно быть вычислено с числом, превышающим модуль , чтобы избежать неопределенности.  
Простой способ достичь этого заключается в вычислении по модулю

|  |  |
| --- | --- |
|  | (29) |

Этот метод избегает неоднозначности при условии или эквивалентно , что легко достижимо на практике. Теперь, уравнение (29) очевидно требует вычисления , однако выражение может быть переписано таким образом, что нужна только четность .

|  |  |
| --- | --- |
|  | (30) |

Функция ранга, , определяется Китайской теоремой об остатках из 8, как:

|  |  |
| --- | --- |
|  | (31) |

Таким образом, четность функции ранга определяется следующим образом:

|  |  |
| --- | --- |
|  | (32) |

где обозначает четность числа , а и все модули считаются нечетными. (Если четное, уравнение (32) неопределено). Этот расчет четности функции ранга может выполняться параллельно с предложенным алгоритмом масштабирования, так что (теперь вычисляемое над конечным полем появляется в той же точке в алгоритме, что и раньше. Пример поможет прояснить ситуацию.

**3.3 Отработанный пример метода однозначного масштабирования**

Число должно быть масштабировано на 2717 для получения , с использованием RNS по модулям , но в этот раз также с использованием . Значения равны , как и раньше. В RNS представлено . Сначала вычислите по модулям 7, 17 и 23, а затем по модулям 11, 13 и 19, используя (15) и (17).

По формулам (30) и (32) можно вычислить параллельно.

Наконец, добавьте к значениям , чтобы получить оставшиеся масштабированные модули:

Таким образом, RNS-значение (или в формате RNS) после масштабирования на 2717 равно . Мы можем проверить результат, преобразовав обратно в десятичное число с помощью Китайской теоремы об остатках:  
   
 В предыдущем примере, вычислялся как 0, так как он был равен , что привело к ошибке в итоговом масштабированном результате. Чтобы поддерживать , требуется всего лишь один дополнительный XOR шлюз или одну дополнительную AND шлюз для каждого сложения или умножения соответственно в остальной части аппаратуры RNS. Очевидно, что это представляет собой незначительные накладные расходы.

**4. Итоги и будущая работа**

В данной статье был представлен новый метод масштабирования чисел RNS путем извлечения ядра числа RNS внутри RNS. Фактически, было извлечено две функции ядра Акушского, а разница между ними использовалась для получения масштабированного результата. Простая техника, требующая поддержки бита четности, показала свою эффективность в устранении ошибок, связанных с неоднозначностью вычисления разности между парами функций ядра Акушского.  
 В дальнейшем можно провести работы, чтобы сделать метод масштабирования более гибким: например, может оказаться возможным масштабирование по формуле . В альтернативе можно попробовать использовать алгоритмы масштабирования с использованием более чем двух функций ядра Акушского. Интересными также будут работы, изучающие другие применения вычисления функции ядра Акушского для RNS: например, базовое расширение становится намного проще, если известно. Наконец, будет интересно использовать данный метод в типичном приложении (цифровая обработка сигналов или криптография с большой длиной слова) для оценки преимуществ в производительности.

**5. Примечания**

[1] “RNS arithmetic: Modern applications in DSP”, ed. M. Soderstrom et al, IEEE Press, 1986

[2] S.R. Barraclough et al: “The Design and Implementation of the IMS A110 Image and Signal Processor”, Proc. IEEE CICC, San Diego, May 1989, pp. 24.5/1-4

[3] A.A. Hiasat and H.S. Zohdy, “Design and implementation of an RNS division algorithm”, Proc. 13th IEEE Symposium on Computer Arithmetic, Asilomar CA, June 1997, pp. 240-249

[4] A.P. Preethy and D. Radhakrishnan, “A 36-bit balanced moduli MAC architecture”, Proc. 42nd IEEE Midwest Symp. Circuits & Systems, Las Cruces NM, August 1999, pp. 380-383

[5] H.A. Al-Twaijry and M. J. Flynn, “Technology scaling effects in multipliers”, IEEE Trans., vol. 47, pp. 1201-1215 (Nov. 1998)

[6] L. Maltar et al, “Implementation of RNS addition and RNS multiplication into FPGAs”, Proc. IEEE FCCM, Napa Valley CA, April 1998, pp. 331-332

[7] M. Re, A. Nannarelli, G.C. Cardarilli, R. Lojacono, “FPGA realization of RNS to binary signed conversion architecture”, Proc. IEEE International Symposium on Circuits and Systems (ISCAS), Sydney, May 2001, pp. 350-353

[8] M.N Mahesh and M. Mehendale, “Low power realization of residue number system based FIR filters”, 13th Int. Conf. on VLSI Design, Bangalore, India, January 2000, pp. 30-33

[9] M. Bhardwaj and B. Ljusanin, “The Renaissance − a residue number system based vector co-processor”, Proc. 32nd Asilomar Conference on Signals, Systems & Computers, Asilomar CA, November 1998, pp. 202-207

[10] N. Burgess, “Scaled and unscaled residue number system to binary conversion techniques using the core function”, Proc. 13th IEEE Symposium on Computer Arithmetic, Asilomar CA, June 1997, pp. 250-257

[11] D.D. Miller et al: “Analysis of the residue class core function of Akushskii, Burcev and Pak”, in “RNS arithmetic: Modern applications in DSP”, op. cit.

[12] F.J. Taylor and C. Huang, "An autoscale residue multiplier", IEEE Trans. Comp., vol. C-31, pp. 321-325 (April 1982)

[13] G. Jullien, "Residue number scaling and other operations using ROM arrays", IEEE Trans., vol. C-27, pp. 325-336 (April 1978)

[14] A. Shenoy and R. Kumaseran, "A fast and accurate RNS scaling technique for high speed signal processing", IEEE Trans., vol. ASSP-37, pp. 929-937 (June 1989)

[15] F. Barsi and M.C. Pinotti, "Fast base extension and precise scaling in RNS for look-up table implementations", IEEE Trans., vol. SP-43, pp.2427-2430 (Oct. 1995)

[16] Z.D. Ulman and M. Czyzak, "Highly parallel, fast scaling of numbers in nonredundant residue arithmetic", IEEE Trans. Sig. Proc., vol. SP-46, pp.487-496 (Feb. 1998)