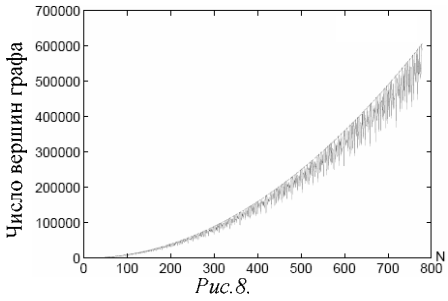

Рис.8.

Увеличение размерности QuIDD-представления оператора обратного преобразования Фурье для $N=2^n$ квантовых бит.

На основании проведённой исследовательской работы производится построение модели квантового вычислителя по методике QuIDD. Начальное состояние моделируемой системы задаётся вводом пси-функций кубитов системы, далее преобразуемых в QuIDD-граф пси-функции системы. Эволюция системы определяется квантовой схемой, набираемой из примитивов в редакторе схем. Подразумевается использование шаблонных QuIDD-графов, где это возможно и их синтез по выведенной методике, в специфических частях схемы. Весь процесс преобразований протекает в терминах графов; результирующая пси-функция транслируется в матричный вид. Помимо результирующей пси-функции, на выходе моделирующей системы предусматривается получение диаграммы временных затрат на моделирование вычисления. Согласно результатам [1], использование методики QuIDD существенно сокращает временные затраты и значительно ускоряет получение решения, что позволит создать высокоэффективную систему моделирования работы квантового вычислителя.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Квантовый компьютер и квантовые вычисления // Журнал “Регулярная и хаотическая динамика”, под ред. *Садовничего В.А.* – Ижевск:, 1999. – 262 с.
2. *Риффель Э., Полак В.* Основы квантовых вычислений // Журнал “Квантовый компьютер и квантовые вычисления ”. – М., 2000, т. 1, № 1.
3. *Viamontes G. F., Markov I. L., Hayes J. P.* High-Performance QuIDD-based Simulation of Quantum Circuits. / http://arXiv:quant-ph/0309060, v2, 29, Nov 2003.
4. *Кокин А.А.* Физические реализации квантового компьютера. М.: ФТИАН / http://elanina.narod.ru/lanina/index.files/student/tehnology/text/kvant.htm#J18
5. *Гузик В.Ф., Гушанский С.М.* Моделирование квантовых схем // Известия ТРТУ. Специальный выпуск. – Таганрог: Изд-во ТРТУ, №9(53), 2005. – С. 66.
6. *Greve David.* QDD: A C++ Quantum Computer Emulation Library. / http://thegreves.com/david/index.html
7. *Chuang Isaac L.* Quantum Computation and Quantum Information. / http://ichuang@mit.edu
8. *Baker Gregory.* Документация к пакету Q-gol / http://www.ifost.org.au/~gregb/q-gol/QgolThesis.pdf
9. *McCubbin Chris.* OpenQUACS / http://userpages.umbc.edu/~cmccub1/quacs/quacs.html
10. *Udrescu Mihai.* Using Hardware Engineering in Quantum Computation: Efficient Circuit Simulation and Reliability Improvement / http://www.sigda.org/daforum/abs/38.pdf

С.Ю. Касаев, В.Е. Золотовский

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ РАНЖИРОВАНИЯ ПРИ ИХ АПАРАТНЫХ РЕАЛИЗАЦИЯХ

При проектировании различных устройств часто возникает проблема сортировки чисел. Это очень длительный и трудоемкий процесс. Существует большое количество различных методов, которые решают данную задачу. Каждый метод

имеет свой алгоритм реализации, который жестко специализирован. Но из-за разнообразия проблем связанных с ранжированием, не существует универсального метода сортировки. В данной статье рассмотрены различные алгоритмы сортировки. Но при аппаратной реализации решения задачи сортировки невозможно использовать все алгоритмы упорядочивания чисел. Это связано с тем, что алгоритмы сортировок разрабатывались при программировании различных задач, и часто используются методы, не доступные при аппаратных решениях (рекурсии, использование процедур, таблиц, циклы без четких границ). Анализ алгоритмов показывает, что их можно разделить на 3 семейства:

1) Просматривается весь список, и очередной элемент вставляется в определенное место – семейство Алгоритмов Вставок.
2) Сравниваются 2 элемента и если необходимо, то они меняются местами – семейство алгоритмов Перестановок.
3) Каждый элемент сравнивается со всеми и его место определяется подсчетом условий – семейство алгоритмов Подсчетов.

Из представленных выше семейств было отобрано несколько алгоритмов. Учитывалась возможность параллельной аппаратной реализации. Алгоритмы, представленные в статье адаптированы для организации параллельных процессов. В статье представлены различные аппаратные реализации, а также приведены временные и аппаратные затраты следующих алгоритмов: метод подсчета, метод пузырька, пирамидальный поиск

1. Метод подсчета. Принцип данного алгоритма заключается в следующем: выбирается элемент последовательности, производится его сравнение с другими элементами, подсчитывается количество неравенств, в которых данный элемент больше, и число таких неравенств будет равняться номеру элемента в упорядоченный последовательности. Этот метод возможно реализовать аппаратно.

Пример последовательной реализации метода подсчета представлен на рис.1. При такой аппаратной реализации требуется стек, в котором содержится неупорядоченная последовательность, схема сравнения, сумматор и регистр накопления. Принцип работы схемы сравнения описывается следующей системой:

$$O = \begin{cases} 1, \text{ если } A>B \\ 0, \text{ если } A<B \end{cases}$$

где A и B – сравниваемые числа; O – признак сравнения.

Принцип работы такой аппаратной реализации заключается в следующем. Последовательно сравнивается 1 элемент со всеми элементами. На сумматор последовательно подаются признаки сравнения. В регистре накапливается сумма признаков. После того, как первый элемент сравнится с остальными элементами, в регистре сформируется адрес первого элемента в упорядоченной последовательности. Далее определяется адрес второго элемента, третьего и так далее. При такой аппаратной реализации время, затрачиваемое на всю сортировку, будет высчитываться по следующей формуле:

$$T_{об} = n\ (n-1)t_i, \qquad (1)$$

где t_i - время, затрачиваемое на 1 сравнение.

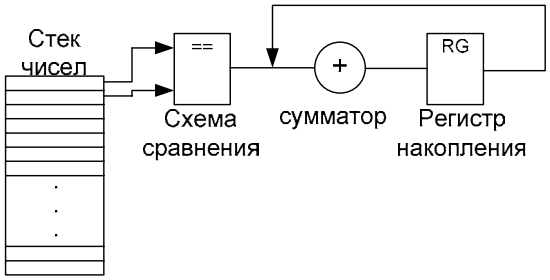

Рис.1. Метод подсчета, последовательная реализация

При такой реализации все элементы сравниваются со всеми. Если ввести запоминание признаков предыдущих сравнений, то возможно ускорить процесс ранжирования. Нет необходимости проводить симметричные сравнения, например, $A_{n-1} > A_n, A_n > A_{n-1}$. Достаточно произвести 1 сравнение, проинвертировать признак сравнения, и получится признак сравнения вторго сравнения. Пример аппаратной реализации приведен на рис.2.

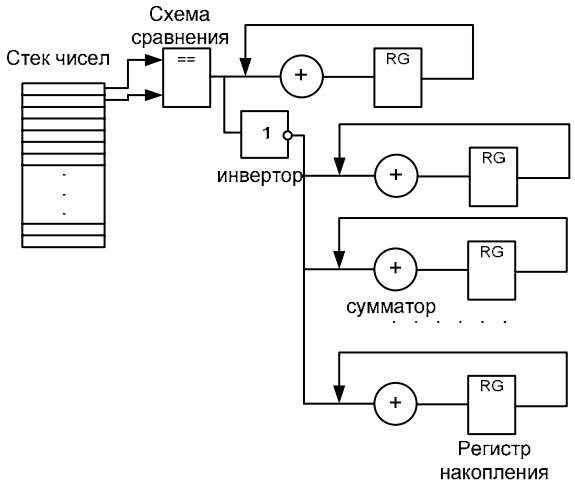

Рис.2. Метод подсчета, модернизированная реализация

При такой реализации количество сравнений с каждым подсчетом будет уменьшаться на 1. Время, затрачиваемое на всю сортировку, определяется по следующей формуле:

$$T_{об} = \frac{n\ (n-1)}{2} t_i . \qquad (2)$$

При введении еще одной схемы сравнения и сумматора время, затрачиваемое на сортировку в реализации (см. рис.1), сократится вдвое. Минимальное время достигается введением n схем сравнения и n сумматоров. Такая реализация приведена на рис.3.

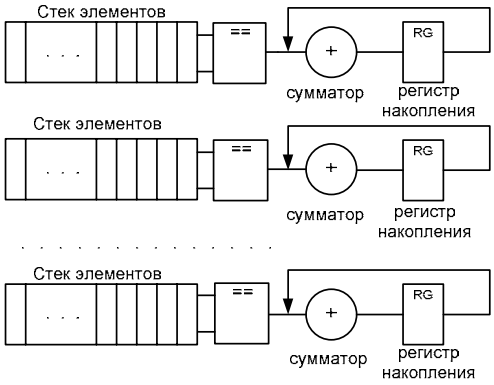

Рис.3. Метод подсчета, параллельная реализация

При такой реализации после первого подсчета в регистрах сформируются адреса элементов в упорядоченной последовательности. Время, затрачиваемое на всю сортировку определяется по формуле:

$$T_{об} = (n-1)t_i . \tag{3}$$

Введение дополнительных аппаратных затрат не ускорит процесс сортировки.

Вывод: анализ метода подсчета показал, что минимальное время сортировки достигается путем введения n- схем сравнения и n-сумматоров.

Метод Пузырька. Принцип данного алгоритма заключается в следующем. Элемент множества сравнивается со следующем, стоящим рядом, если следующий элемент больше предыдущего, то происходит перестановка. Следуя этому принципу, количество сравнений с каждым шагом уменьшается на 1. Общее количество сравнений:

$$N = \frac{n(n-1)}{2}, \tag{4}$$

Данный метод можно реализовать аппаратно, используя всего одну схему сравнения. Такая реализация представлена на рис.4.

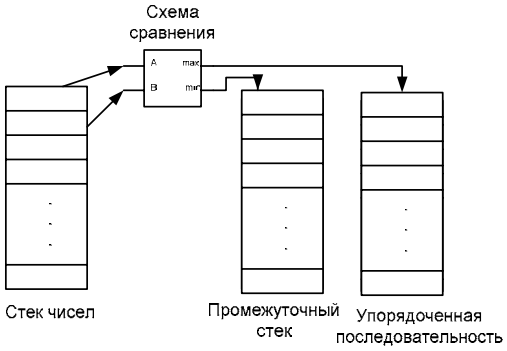

Рис.4. Метод Пузырька, последовательная реализация

Принцип работы данной аппаратной реализации заключается в следующем. Сравниваются 2 элемента, выбирается максимум, минимум помещается в промежуточный стек. После полного прохода выбирается максимальный элемент, помещается в стек упорядоченной последовательности. Из промежуточного стека эле-

менты записываются в основной стек и сравнение повторяется. Время выполнения сравнения при такой реализации высчитывается по формуле:

$$T_{об} = N \times t_i = \frac{n(n-1)}{2} t_i . \quad (5)$$

При i-ом количестве схем сравнения время выполнения будет сокращаться в i раз:

$$T_{об} = \frac{n\ (n-1)}{2 \cdot i} t_i\ . \quad (6)$$

Параллельная аппаратная реализация представлена на рис.5.

Время выполнения операции сравнения будет высчитываться по формуле (6). При аппаратной реализации возможно ввести параллельность сравнений – нет необходимости ждать выполнения всего первого прохода чтоб сравнить первых два элемента промежуточной последовательности. Задержка модернизированного способа высчитывается по формуле:

$$T_{об} = (n-1+n-2) \times t_i = (2n-3) t_i . \quad (7)$$

Вывод: анализ метода пузырька показал, что самое минимальное время выполнения операции сортировки составит $T_{об} = (n-1) t_i$, при условии, что в аппаратной реализации используется n схем сравнения (ри.6).

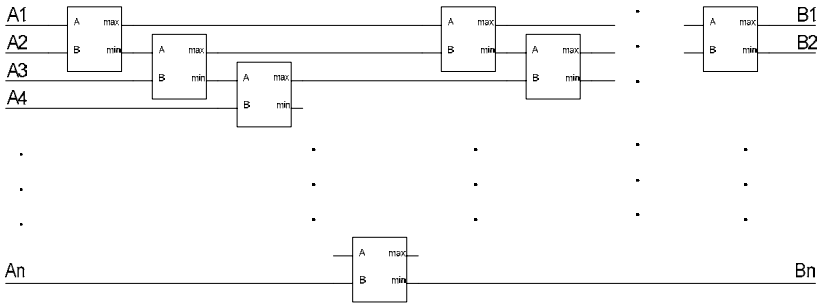

Рис.5. Метод Пузырька, параллельная реализация

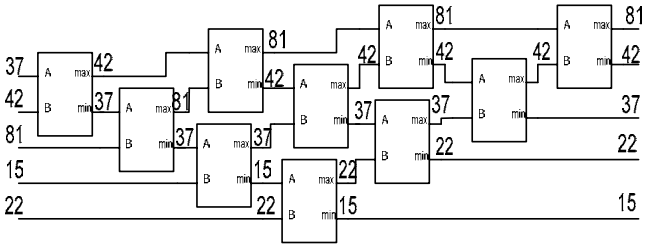

Рис.6. Метод Пузырька, модернизированная реализация

Пирамидальная сортировка. Идея данного метода заключается в разбивании всех элементов на пары, выбираются максимумы, затем максимумы разбиваются на пары и среди них находятся максимумы и т.д. В конце первого прохода находится максимум. Затем в первоначальном множестве данный элемент помечается как минимальный и процесс повторяется. В общем случае время сортировки определяется по формуле:

$$T_{об} = N \cdot t_i \cdot \log_2 N \ . \qquad (8)$$

Возможны последовательная и параллельная реализация данного метода. При параллельной реализации сравнение чисел происходит параллельно. При последовательной – используется память для хранения промежуточных максимумов.

Вывод: данный метод удобно использовать при поиске максимума, для полного ранжирования этот метод требует достаточно больших аппаратных затрат, показывая при этом долгое время выполнения.

Заключение. Результаты сравнения рассмотренных реализаций алгоритмов сортировок, приведенные в таблице, показывают, что минимальное время достигается при последовательной реализации метода подсчета при количестве схем сравнения, равным количеству элементов в неупорядоченной последовательности.

Таблица

Время ранжирования различных аппаратных реализаций

		количество сравниваемых элементов			
название метода		4	8	16	32
Метод подсчета	Последовательныйс 1 схемой	$12t_i$	$56t_i$	$240t_i$	$992t_i$
	Последовательный 2 схемами	$6t_i$	$28t_i$	$120t_i$	$446t_i$
	Последовательный n-схемами	$3t_i$	$7t_i$	$15t_i$	$31t_i$
Метод пузырька	Последовательный с 1 схемой	$6t_i$	$28t_i$	$120t_i$	$446t_i$
	Последовательный 2 схемами	$4t_i$	$14t_i$	$60t_i$	$223t_i$
	Параллельный	$6t_i$	$28t_i$	$120t_i$	$446t_i$
	Модернизированнный	$5t_i$	$13t_i$	$29t_i$	$61t_i$
Пирами-дальная сортировка	Последовательный с 1 схемой	$12t_i$	$48t_i$	$240t_i$	$992t_i$
	Последовательный n схемами	$8t_i$	$24t_i$	$64t_i$	$160t_i$
	Параллельный	$6t_i$	$20t_i$	$56t_i$	$155t_i$

Ю.В. Чернухин, А.В. Лунев

НЕЙРОСЕТЕВОЙ ПОДХОД К ВЕКТОРНОМУ КВАНТОВАНИЮ КОЭФФИЦИЕНТОВ ОТРАЖЕНИЯ ПРИ ВОКОДИРОВАНИИМЕТОДОМ ЛИНЕЙНОГО ПРЕДСКАЗАНИЯ

Введение. В настоящее время эффективное сжатие речевых данных (вокодирование) является весьма важной задачей в области современных систем телекоммуникаций. В большинстве существующих алгоритмов вокодирования одним из этапов является векторное квантование первичных признаков при анализе речи [1]. Эффективность квантования таких признаков непосредственно сказывается как на качестве воспроизводимой речи, так и на скорости ее передачи. Оба эти параметра являются основополагающими для вокодеров.

Традиционно векторное квантование выполняется на основе кодовой книги [1], которая формируется на этапе проектирования. Наиболее распространен-