1

УДК 06.055.2 + 061.3 + 002.2 + 070.43

РАЗРАБОТКА УСТРОЙСТВА АППАРАТНОЙ СОРТИРОВКИ С ПОСТОЯННЫМ ВРЕМЕНЕМ ВВОДА И ВЫВОДА

Зинченко Ю.Е, Шерекин А.А.
Донецкий национальный технический университет

В статье рассматриваются варианты аппаратной реализации сортирующих устройств на базе различных структур данных. Выполняется анализ четырех элементарных вариантов сортирующих устройств, раскрываются их преимущества, недостатки и делается попытка на их базе разработать сортировщик, оптимальный со стороны производительности и аппаратных затрат

Введение

Проблема сортировки данных встречается во всех областях, связанных с компьютерной техникой. Об актуальности данной проблемы говорит многочисленное количество уже существующих алгоритмов сортировки: сортировка пузырьком, сортировка вставками, перемешиванием, быстрая сортировка. Каждый из алгоритмов имеет свои преимущества и недостатки в зависимости от сложности алгоритма, времени, необходимого на сортировку, и потреблению памяти для выполнения сортировочных действий. Это касается программной реализации. Если же взглянуть на аппаратную сортировку, то здесь также найдется большое количество всевозможных алгоритмов. Основным различием в них является используемая структура данных. Это может быть FIFO-массив [2], бинарное дерево [2], различные Heap-структуры [1], сдвиговые регистры [2], систолические массивы [3, 4]. Некоторые из алгоритмов предполагают наличие кроме структуры, хранящей данные, дополнительное устройство, управляющее ее работой – менеджер сортировщика, что предполагает дополнительные аппаратные затраты.

Одной из областей применения аппаратных сортирующих устройств являются компьютерные сети, а именно маршрутизаторы. При прохождении через маршрутизатор пакеты от различных пользователей имеют свой приоритет на обработку. Если алгоритм сортировки плох и работает медленно, это может нивелировать преимущества высокоскоростного подключения, будут образовываться очереди в маршрутизаторе, что также может привести к дальнейшей потере пакетов, вследствие конечности очереди внутри маршрутизатора. Если алгоритм быстр, но требует для такого быстродействия слишком много аппаратных затрат, то это скажется на стоимости устройства и на его конкурентоспособности на рынке.

В данной статье рассматриваются 4 варианта реализации приоритетных сортирующих очередей. Критерием по выбору алгоритмов является постоянное время добавления нового элемента в очередь и получения наиболее приоритетного элемента из очереди. После анализа предложенных алгоритмов, рассматривается вариант реализации на основе двух из них, включающий в себя положительные черты каждого и максимально преуменьшающий их недостатки.

Алгоритм сортировки с использованием бинарного дерева схем сравнения

Структура данной приоритетной очереди включает N элементов памяти, где N – максимальное количество элементов, которые способно отсортировать данное устройство, и бинарное дерево схем сравнения, глубиной $\log_2 N$. Каждый элемент дерева представляет собой схему сравнения двух значений, на выход которой подается большее или меньшее из них, в зависимости от того, какое численное значение приоритета – большее или меньшее – считается более приоритетным в реализуемой системе. Кроме того в данной структуре предусмотрена обратная связь, позволяющая удалить из очереди элемент, выданный на выход по запросу извлечения. Схема приоритетной очереди на базе бинарного дерева приведена на рис.1

Недостатком данного алгоритма является отсутствие поддержки принципа FIFO. Кроме

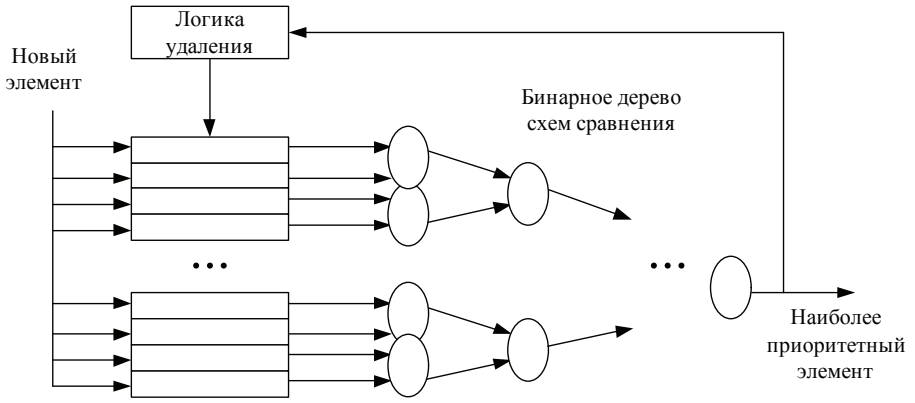

Рисунок 1. Структура сортировщика с использованием бинарного дерева

того проблематичной является расширяемость очереди. При добавлении элементов памяти будет увеличиваться необходимое количество схем сравнения и глубина бинарного дерева, что приведет к растущей задержке в работе устройства.

Алгоритм сортировки с использованием FIFO-массива

Структура данного сортировщика приведена на рис.2

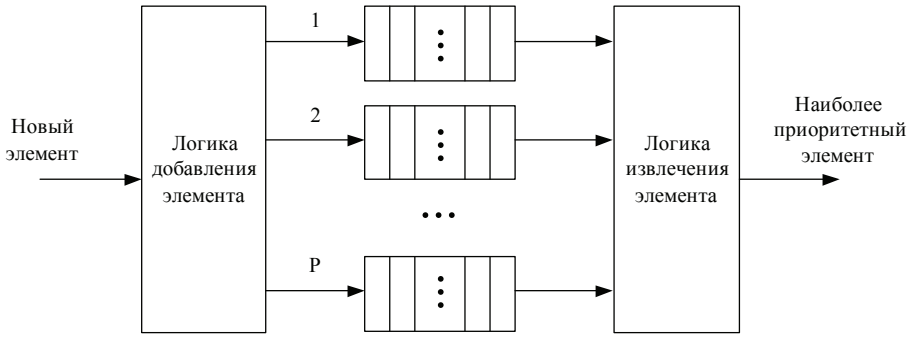

Рисунок 2. Структура сортировщика с использованием FIFO-массива

Данная структура условно делится на 3 части: логика добавления нового значения, P очередей по N элементов в каждой, где P – количество поддерживаемых приоритетов, а N – максимальная длина очереди, и логика выдачи очередного элемента. Первый блок выполняет распределение входящего элемента в зависимости от его приоритета в одну из P очередей. В каждой очереди находятся одноприоритетные элементы. В данной реализации поддерживается принцип FIFO. Третья часть структуры выполняет последовательный анализ очередей в порядке убывания приоритетов. Сначала выдаются элементы из наиболее приоритетной очереди. Если она пуста, то анализируется следующая по степени приоритетности очередь и т.д.

Явным недостатком данного алгоритма являются чрезмерные аппаратные затраты. Так, если в какой-то момент времени будут приходить пакеты одного приоритета, то только одна очередь будет использоваться, в то время как остальные будут простаивать. Проблема расширяемости также актуальна для данной реализации. Для добавления количества поддерживаемых приоритетов необходимо добавить очередь, длиной N, т.е. дополнительно N элементов памяти. Для расширения максимальной длины очереди, необходимо во все P очередей добавить по одному элементу памяти.

Алгоритм сортировки с использованием сдвигового регистра

Структура сортировщика приведена на рис.3.

В структуре данной реализации имеется N сдвиговых регистров, которые постоянно поддерживают упорядоченное состояние очереди. Каждый регистр соединен с соседним, и на каждый

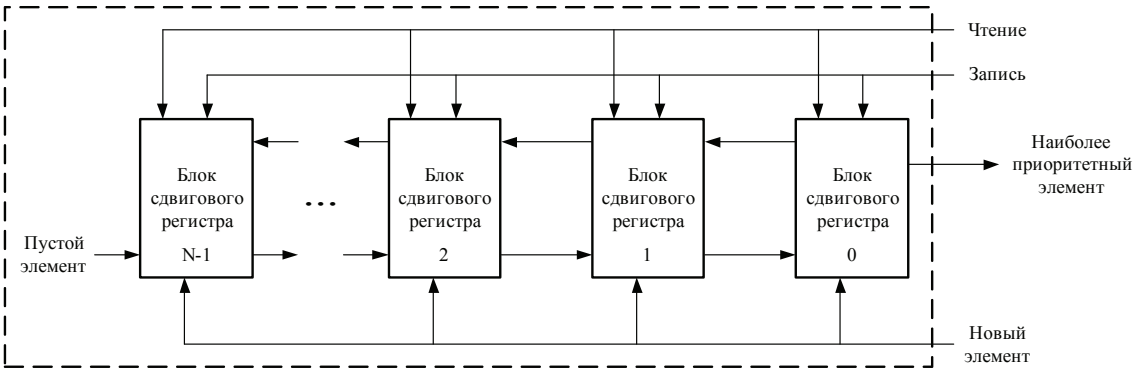

Рисунок 3. Структура сортировщика на базе сдвигового регистра

регистр поступает добавляемый элемент. По причине поддержания постоянной упорядоченности при добавлении нового элемента существует только 1 позиция, доступная для него – после всех элементов, имеющих больший либо равный приоритет (нестрогость условия сравнения обусловлена обеспечением принципа FIFO). Таким образом, более приоритетные элементы всегда будут находиться ближе к выходу, чем менее приоритетные.

Каждый блок сдвигового регистра состоит из регистра, хранящего значение, схемы сравнения нового значения с текущим, и логики, анализирующей результат сравнения в предыдущем блоке. Результат работы схемы сравнения передается следующему блоку, чтобы тот мог определить, что необходимо сделать с новым элементом. Возможны 3 варианта действий при добавлении нового элемента в очередь: запись нового элемента в регистр, игнорирование нового элемента и запись значения предыдущего регистра. При извлечении элемента выполняется сдвиг всех значений блоков на один по направлению к выходу. Структура блока сдвигового регистра приведена на рис.4.

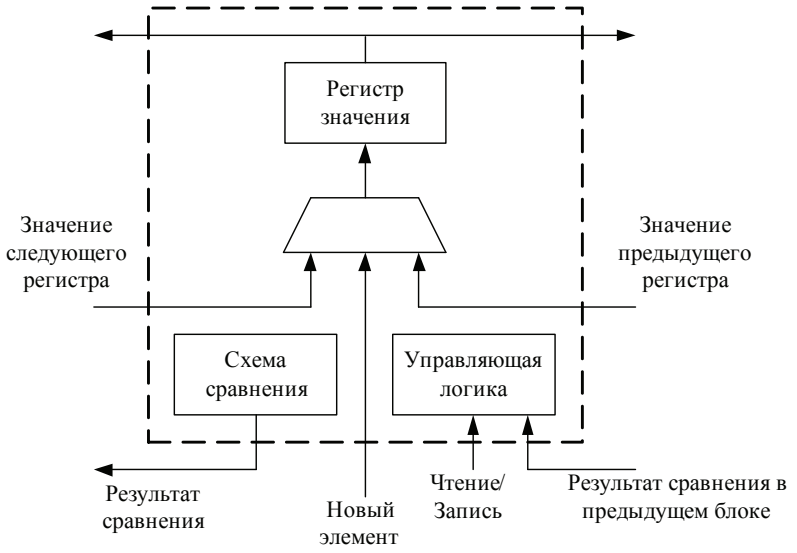

Рисунок 4. Структура блока сдвигового регистра

Положительными сторонами данной реализации является хорошие показатели по критерию аппаратных затрат – для каждого блока регистра сдвига необходима лишь одна ячейка памяти. Свободная расширяемость также является важной особенностью данного алгоритма. Для увеличения максимальной длины очереди необходимо лишь подключить к концу одного регистра сдвига другой. Никакой дополнительной логики не требуется. Из недостатков же выделяется проблема загрузки шины. При относительно небольшом количестве элементов сортировщик работает на максимально возможной скорости, но при постоянном увеличении длины очереди из-за того, что вход, на который подается новый элемент, связан с каждым блоком сдвигового регистра, максимальная частота надежной работы устройства постепенно падает.

Алгоритм сортировки с использованием систолического массива

Структура элементов данного алгоритма несколько похожа на метод со сдвиговым регистром. В состав массива входят систолические блоки, внутри которых происходит сортировка данных. Блоки соединены между собой в массив. Структура систолического массива приведена на рис.5.

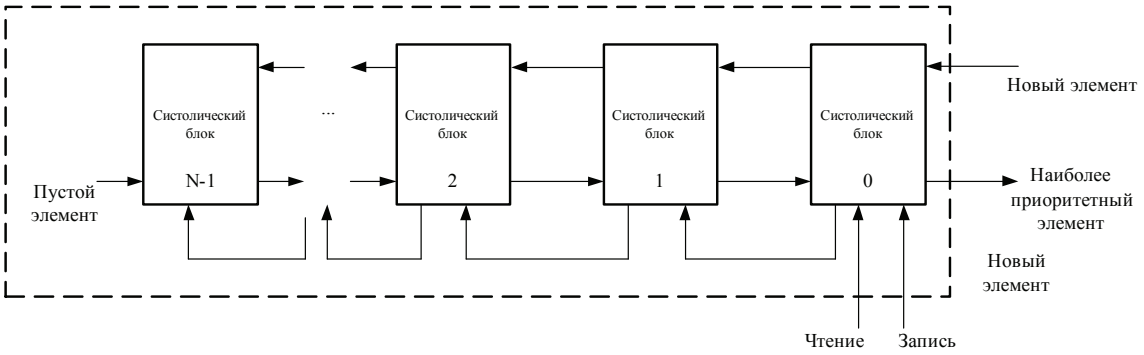

Рисунок 5. Структура сортировщика на базе систолического массива

Каждый блок массива содержит 2 регистра: регистр значения и временный регистр, схемы сравнения и управляющую логику. В каждом цикле добавления нового элемента выполняется сравнение его со значением текущего блока. Структура блока систолического массива приведена на рис.6.

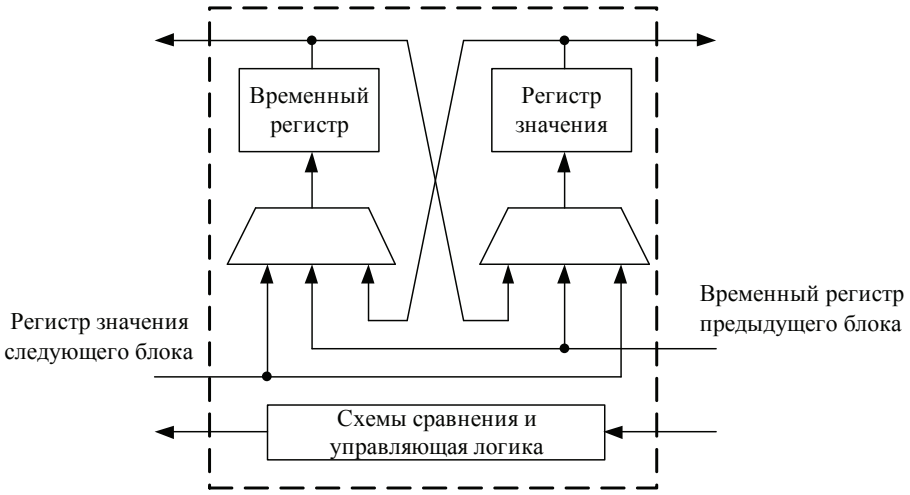

Рисунок 6. Структура блока систолического массива

Через определенное количество тактов весь массив будет отсортирован. Не смотря на то, что для полной упорядоченности элементов систолического массива необходимо время, на каждом шаге работы устройства наиболее приоритетный элемент будет находиться в первом блоке, что позволяет в любое время выполнить извлечение и получить ожидаемое значение. При извлечении элемента данные в каждый регистр блока поступают из предыдущего блока систолического массива.

Проблемой данной реализации является несоблюдение принципов FIFO при выдаче информации. Для решения проблемы каждый временный регистр блока снабжается дополнительным битом приоритетности. Если на очередном шаге записи элемента в блок во временный регистр были переданы данные со входа, а не из регистра значения, то этот дополнительный бит равен 0, иначе – 1. При передаче элемента из блока в блок, как при вставке нового элемента, так и при извлечении, данный бит участвует в сравнении приоритетов наряду с полем приоритета.

Еще одной негативной чертой данной реализации является необходимость использования по две ячейки памяти для каждого элемента очереди.

Из достоинств можно привести легкую расширяемость, подобно сортировщику на сдвиговых

регистрах, и отсутствие проблемы загрузки шины, так как данные передаются из блока в блок последовательно.

Проанализировав 4 возможных структуры приоритетной очереди, можно сделать вывод, что 2 последние имеют преимущество над остальными. Недостатки же каждой из них можно нивелировать за счет достоинств другой. Так, если в структуру сортировщика на базе систолического массива вместо одного регистра значения поместить сдвиговый регистр небольшого размера, то возможно решение сразу 2 проблем – проблемы загрузки шины в сдвиговом регистре (не будет необходимости в большой длине сдвигового массива) и проблемы аппаратных затрат систолического массива (так как количество блоков уменьшится за счет сдвигового регистра).

В дальнейшем планируется выполнить анализ смешанной структуры приоритетной очереди с целью оптимизации аппаратных затрат устройства, при условии заданного минимального быстродействия, а также определения максимально возможной скорости работы, при ограниченной площади кристалла.

Выводы

Аппаратная сортировка является узким местом во многих устройствах, в которых она применяется, поэтому оптимизация процесса сортировки может существенно улучшить скоростные характеристики устройства. Но увеличение скорости работы, сопровождаемое чрезмерными аппаратными затратами, не приведет к желаемому результату, вследствие удорожания устройства. Попытка найти алгоритм реализации, компромиссный с обеих точек зрения, была выполнена в данной статье.

Литература

[1] Bhagwan R., Lin B. Design of a High-Speed Packet Switch with Fine-Grained. Quality-of-Service Guarantees / R. Bhagwan, Lin B. // IEEE International Conference on Communications, New Orleans, USA, 18-22 June 2000. – New Orleans, 2000. - P. 1430-1434.

[2] Moon S.W., Rexford J., Shin K.G. Scalable Hardware Priority Queue Architectures for High-Speed Packet Switches / S.W. Moon, J. Rexford, K.G. Shin // IEEE Transactions on Computers, Washington, DC, USA, Nov. 2000 – Washington, 2000. – P. 1215-1227.

[3] Bednara M., Beyer O., Teich J., Wanka R. Hardware-Supported Sorting: Design and Tradeoff Analysis / M. Bednara, O. Beyer, J. Teich, R. Wanka // 3rd Workshop on System Design Automation. – P. 37-44.

[4] Leiserson C.E. Systolic Priority Queues / C.E. Leiserson, Department of Computer Science Carngie-Mellon University Pittsburgh, Pennsylvania, April 1979.