Конфигурируемые процессоры для визуализации биомедицинских данных[1]

Часто при решении задач цифровой обработки сигналов ставятся жесткие требования к производительности компьютера. Например, желательно выделять полезную информацию из видеосигнала в момент его ввода, избегая решений, когда сигнал сначала сохраняется, а затем обрабатывается процессором. Обычно производительности процессора не хватает, и алгоритм обработки сигнала упрощается программистами до такой степени, что он теряет свою эффективность. Другой выход из положения — использование специализированного процессора, который умеет выполнять только один алгоритм. Но в этом случае страдают те исследователи, которые хотят использовать разные алгоритмы для обработки сигнала с разными параметрами. Естественное решение в таком случае — синтезировать процессор непосредственно перед выполнением алгоритма, который будет работать в системе, не имеющей заранее известной архитектуры. Способность выполнить алгоритм за короткое время — основное отличие конфигурируемого процессора от классического. Основные требования предъявляемые к такому процессору — способность выполнить заданный алгоритм в течении заданного времени, использовав при этом определенные вычислительные ресурсы. Предположим, что наш процессор удовлетворяет таким требованиям и назовем эти требования параметрами процессора. Тогда при синтезе такого процессора мы увидим, что между параметрами существует простая взаимосвязь. Вот пример.

Существует два базовых способа вычислений — комбинаторный и рекурсивный. Операция умножения выполняется первым способом за один такт времени — просто все результаты запоминаются в памяти и быстро воспроизводятся при появлении операндов. Рекурсивный способ умножения похож на умножение столбиком — для этого надо много раз сложить и сдвинуть операнды, чтобы получить результат. Решая одну и ту же задачу в первом процессоре мы выигрываем во времени но тратит большие вычислительные ресурсы, во втором процессоре — выигрываем ресурсы, но теряем время. Здесь рекурсивный способ вычисления отождествляется с вычислением с помощью машины Тьюринга, а комбинаторный с вычислением с помощью просмотровой таблицы.

Каждое реальное вычислительное устройство можно условно отоб-

[1]"Биосистемы в экстремальных условия", Вычислительный центр РАН, М., 1996. С. 47– 50 Работа выполнена при поддержке гранта РФФИ 95-01-01623

разить в точку на континууме между комбинаторным и рекурсивным вычислителями. Сдвигая эту точку в ту или иную сторону мы получаем процессор с нужными характеристиками. Не всегда возможно получить желаемый процессор, и тогда решается задача оптимизации с параметрами — сложностью алгоритма, временем обработки данных, требуемыми вычислительными ресурсами.

Конфигурируемый процессор можно представить как многомерный массив ячеек, связанный локальными и глобальными каналами. Локальный каналы образуют общую структуру процессора, а глобальные служат для обмена между отдельными логическими модулями и для обмена с внешним миром. Ячейка состоит из просмотровой таблицы, в которую можно загрузить комбинаторную функцию, и регистра с обратной связью для выполнения рекурсивной функции. Основной идеей является то, что каждая ячейка работает как самостоятельная машина Тьюринга. Она может сдвигать данные в другие ячейки и эти данные являются обрабатываемыми данными для этой ячейки. Но и другие ячейки могут сдвигать в нее данные — для нашей ячейки они являются командами и изменяют ее состояние и функции, которые она выполняла до этого. Таким образом конфигурация процессора и обработка данных могут происходить одновременно. Ячейки способны подключаться через локальные каналы друг к другу, образовывать структуры, занимать глобальные каналы, моментально принимая данные из удаленных структур или извне процессора и так далее. Нужно отметить, что такой процессор отличается от существующих программируемых логических микросхем тем, что последние программируются только один раз перед выполнением алгоритма, их макроячейки выполняют одну заранее известную функцию и не могут изменять коммутацию друг с другом.

Начальное конфигурирование процессора и управление всеми его последующими действиями происходит следующим образом. Алгоритм обработки сигнала формализуется и описывается на языке, похожим на параллельный C++. Компилятор этого языка аналогичен кремниевым компиляторам. Алгоритм представляется машиной состояний и транслируется в набор логических примитивов. Основное различие здесь состоит в том, что кремниевые компиляторы делают декомпозицию схемы, а в нашем случае необходимо делать и композицию и декомпозицию одновременно. При этом (в соответствии в вышеописанной задачей оптимизации) создаются комбинаторные и рекурсивные узлы, работающие на разных логических уровнях. Например современную вычислительную систему можно описать в наших терминах так: Несколько одновременно работающих процессоров решает задачу и это комбинаторный вычисли-

тель верхнего уровня. Каждый процессор является машиной состояний, и его последующее состояние зависит от предыдущего — это рекурсивный вычислитель более низкого уровня. В процессоре работает микропрограмма, управляющая электронными блоками — комбинаторными вычислителями более низкого уровня. Каждый блок — это какой-либо вычислитель. Например это умножитель, любого из типов, описанных в этом тексте выше.

Таким образом весь процессор организуется в логическую структуру, которая имеет свою иерархию и на каждом ее уровне работают машины состояний, включающие в себя некоторые блоки. Эти блоки можно назвать вычислителями имеющими ту или иную степень рекурсивности.

Конечно, сейчас конфигурируемые процессоры не является универсальными из-за технологических ограничений, но тем не менее они очень удобны при создании систем, где время обработки или ресурсы являются критическими параметрами. Основные области применения — системы быстрого реагирования, синтаксические разборщики, системы распознавания. Очень удобны такие процессоры при адаптивной фильтрации, причем под адаптивностью здесь понимается не изменение параметров фильтра в зависимости от входных данных, а изменение самого алгоритма фильтрации которое может быть как скачком от одного алгоритма к другому, так и непрерывным преобразованием.