Ключевые особенности рекуррентной архитектуры и возможные пути ее развития

Волчек В.Н. ИПИ РАН

Аннотация: *В статье рассматриваются ключевые особенности рекуррентной архитектуры, отличающие её от других традиционных парадигм. Выявляется достоинства и недостатки предложенных решений, обозначаются возможные пути дальнейшего развития.*

Введение

Усовершенствование существующих и поиск новых архитектур вычислительных систем является одной из наиболее актуальных задач в мире компьютерной индустрии. Обеспечение поддержки параллельных вычислений - главный тренд развития современных микропроцессоров. Как уже отмечалось в [1] классическая фон-неймановская архитектура, управляемая потоком команд (control flow), изначально ориентирована на последовательные вычисления и слабо адаптируется под задачи параллелизма.

Гораздо лучше с теоретической точки зрения для параллельных вычислений подходят архитектуры потока данных (data flow) [2]. Однако опыт исследователей последних 35 лет показал, что практическая поддержка потоковой архитектуры является сложной и ресурсоемкой задачей, которая включает:

- необходимость большого объема запоминающей среды;
- частое обращение к памяти, что приводит к значительным временным задержкам;
- ® использование широкополосной пропускной среды для передачи информации по внутренним шинам;
- сложную модель программирования, требующую коренных изменений в сознании программистов;
- сложность и высокую стоимость аппаратуры.

Кроме того, выполнение некоторых типов вычислений, например, строго последовательного вычисления в потоковых архитектурах неэффективно [3].

В результате анализа достоинств и недостатков двух диаметрально различных архитектурных подходов (control flow и data flow) появилось новое направление «гибридных» парадигм (data-control flow), объединяющее в себе обе архитектуры в рамках единой вычислительной системы. Анализ современных зарубежных разработок и исследований «гибридного» направления был представлен на 1-й школе молодых ученых ИПИ РАН в 2010 году [4].

Коллектив отдела перспективных компьютерных систем ИПИ РАН на протяжении последних лет, среди прочего, занимается исследованием рекурретной потоковой архитектуры (РПА), с практической реализацией в форме

1) Рекуррентная потоковая архитектура

четырехядерного цифрового сигнального процессора [5]. Данная парадигма сочетает в себе свойства традиционных фон Неймановских архитектур со свойствами нетрадиционных потоковых, что позволяет отнести РПА к классу «гибридных архитектур».

Ключевыми особенностями данной архитектуры являются:

- объединение потока команд и потока данных в единый поток самодостаточных данных;
- управляющий уровень разрабатываемой архитектуры, управляется потоком команд, как в традиционных фон Неймановских системах;
- поток самодостаточных данных рекуррентно свернут на стадии компиляции, и автоматически разворачивается при помощи специализированного устройства преобразования тегов (УПТ).

Рассмотрим и проанализируем данные особенности более подробно.

2) Единый поток самодостаточных данных

Предлагаемая РПА нетрадиционна и радикально отличается по основным моментам не только от классической парадигмы фон Неймана, но и от других нетрадиционных параллельных парадигм и базирующихся на них архитектур.

Для существующих традиционных и нетрадиционных компьютерных архитектур характерно наличие двух потоков: активного - команд и пассивного - данных.

В РПА оба потока сливаются в один общий поток элементов самодостаточных данных (ЭСД), в котором, помимо собственно обрабатываемых данных, содержится и необходимая для их обработки управляющая и служебная информация (в традиционных архитектурах кодируемая в форме команд). Команда и данное представляет собой единое неделимое целое и совместно перемещается в рамках вычислительного процесса.

Наглядное представление о сравнительных качествах рассматриваемых вычислительных парадигм дает их сопоставление по организации памяти (рис. 1) и по количеству шагов, необходимых для выполнения команды (рис. 2). Рисунки условны и приводятся только для наглядности.

Как видно из рисунка 2 в РПА требуется меньшее (предельно допустимое) количество логических шагов для организации вычислительного

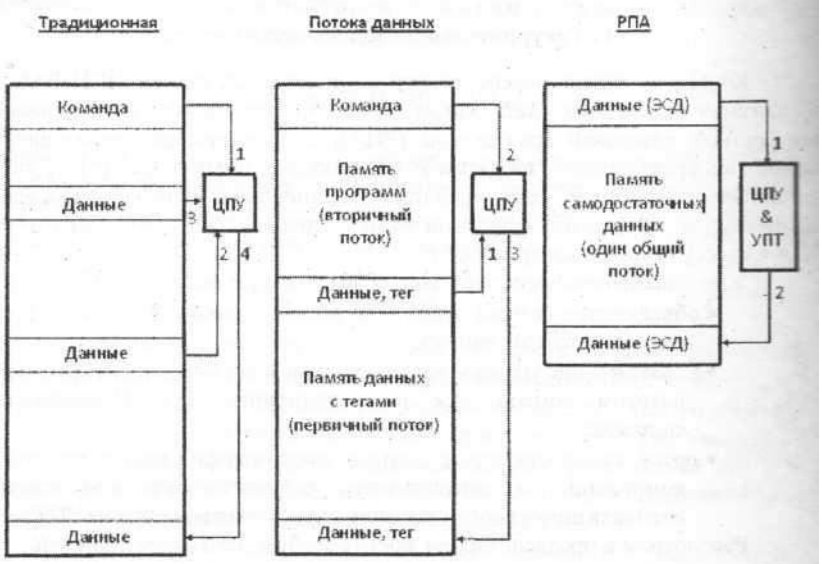

Рис. 1. Принципиальные отличия сравниваемых парадигм

Рис. 2. Выполнение команды в сравниваемых парадигмах

С учетом того, что память в настоящее время является одним из самых медленных устройств, сокращение количества обращений к ней приведет к приросту производительности, а также упростит синхронизацию вычислительного процесса. Как уже говорилось выше, проблема памяти является ключевым камнем преткновения в развитии потоковых архитектур. Поэтому внедрение единого потока самодостаточных данных представляется перспективным для применения в реальном «железе».

Однако подобное решение требует большой пропускной способности коммуникационной сети внутри устройства, что может привести к увеличению размеров кристалла, а также потенциально к повышенному энергопотреблению и необходимости дополнительного охлаждения.

3) Конвейерная организация операционного уровня

В соответствии с количеством логических шагов в архитектуре, в практической реализации цикл выполнения команд может состоять из трех ступеней конвейера Q_b Q_{2j} Q3. Выполнение каждой ступени конвейера совмещено во времени, таким образом на каждом такте происходит обработка текущей пары ЭСД и выборка следующей. Рассмотрим работу конвейера более подробно на примере четырехядерного процессора с РПА.

Как только на вход операционного уровня поступают первые ЭСД на первом такте (Tt) происходит распределение ЭСД по вычислительным ядрам (Qi). В четырехядерной реализации предусматривается одновременное распределение 8 ЭСД (A^). В следующем такте (T_2) в памяти совпадений происходит процесс обнаружения пар ЭСД, поступивших в каждое вычислительное ядро на предыдущем такте (Q_2). Параллельно с процедурой выборки пары, происходит распределение новых поступивших ЭСД. ЭСД уже нашедшие пару на Q_2, помечаются как готовые для исполнения и отсылаются для вычисления результата в следующем такте T_3.

В такте T_3, дополнительно к ступеням конвейера Qi и Q_2, начинает работать ступень конвейера Q_3, на которой происходит вычисление результата и отсылка его в единую коммуникационную шину. Таким образом первый вычислительный цикл становится законченным, получены первые результаты.

T_1	T_2	T_3	T_4	
Распределение ЭСД A_{1-8}	Распределение ЭСД B_{1-8}	Распределение ЭСД C_{1-8}	Распределение ЭСД D_{1-8}, A'_{1-4}	Q_1
	Формирование пар ЭСД A_{1-8}	Формирование пар ЭСД B_{1-8}	Формирование пар ЭСД C_{1-8}	Q_2
		Вычисление результатов A'_{1-4}	Вычисление результатов B'_{1-4}	Q_3

Рис.3. Конвейерная организация операционного уровня

Полученные на Q_3 результаты могут быть переданы на управляющий уровень, либо могут быть заново вовлечены в вычислительный процесс. В случае если необходимо результаты А’м вовлечь в вычислительный процесс,

результаты A'_{1-4} поступают на ступень конвейера, где происходят распределение поступивших результатов A'_{1-4} и поступивших с управляющего уровня ЭСД D_{1_8}. Так как устройство распределения (распределитель) может распределить только 8 ЭСД по вычислительным ядрам, распределение происходит по приоритетам, установленным в конфигурационных настройках процессора.

Также стоит отметить, что событийно-ориентированная природа **РПА** позволяет реализовать асинхронный конвейер, где каждая ступень работает **по** классическому синхронному типу.

3) Управляющий уровень с традиционной архитектурой, управляемой потоком команд.

Специфика программирования многоядерного рекуррентного потокового процессора скрыта от пользователя традиционным фон- неймановским процессором, который также решает задачи управляющего уровня.

Использование традиционной архитектуры на управляющем уровне:

- обеспечивает совместимость с внешним окружением вычислительной системы;
- позволяет использовать имеющиеся средства разработчика **с** традиционными языками программирования для загрузки вычислительного процесса на операционном уровне;
- позволяет выполнять ряд вспомогательных функций, таких как предобработка входных данных, управление исключительными ситуациями и т.д.;
- позволяет эффективно реализовывать последовательные участки алгоритма.

Стоит отметить, что вопрос совместимости и программирования **в** нетрадиционных архитектурах на протяжении не одного десятка лет представляет собой камень преткновения для массового перехода на параллельные вычисления. Поэтому наличие соответствующего программного обеспечения для поддержки разработчика является ключевым критерием **в** достижении коммерческого успеха предлагаемой архитектуры.

Также, как и в любой другой параллельной системе наличие последовательного информационного потока может проявиться эффектом «бутылочного горлышка», что негативно скажется на обработке больших объемов данных. Для этих целей имеет смысл исследовать возможность реализации кластерной или многопроцессорной системы с РПА, с разным уровнем иерархии.

4) Преобразователь тегов

Другим инновационным свойством, призванным справиться с недостатками dataflow, является способность автоматически вычислять код

операции следующего шага, а также адрес, по которому необходимо отправить результат текущей операции. Поступающая на вход операционного уровня программа, подвергается рекуррентной свертке на фазе компиляции и при помощи специализированного устройства преобразования тегов разворачивается в ходе вычислительного процесса. Данная особенность зжззволяет сократить, как количество обращений к памяти, так и (что особенно важно*)* взаимодействие с управляющими устройствами системы. В идеале, данное решение также способно существенно сократить объемы запоминающей среды.

Теги содержат некоторую начальную сжатую информацию, обеспечивающую выполнение требуемой процедуры. Каждый следующий шаг *процедуры рекуррентно* самоопределяется, в том *числе с* учетом *результата* предыдущего шага.

В *разрабатываемой архитектуре* в состав ЦПУ *включено* автономное устройство преобразования тегов (УПТ), которое обладает возможностью саморазвертки *рекуррентного* вычислительного процесса. УПТ инициируется операндами, пришедшими на обработку в ЦПУ, работает параллельно с ЦПУ и определяет действие на следующем шаге вычислительного процесса «модифицирует тег). УПТ представляет собой относительно простую (по аппаратным затратам) комбинационную схему, содержащую средства настройки (в необходимых случаях) на предметную область.

В памяти нет исполняемой программы в традиционном смысле; есть только начальные значения функциональных полей операндов, которые динамически подвергаются рекуррентной саморазвертке устройством УПТ. Для выполнения алгоритма необходимо задать начальные значения функциональных полей.

Однако на текущий момент опробован и реализован самый простой вид универсального преобразования. Задача создания других специализированных преобразователей тегов представляет собой серьезный научный интерес и может быть развита в дальнейшем. В частности, с учетом распространения реконфигурируемых средств вычислительной техники на базе ПЛИС, существует возможность перепрограммировать УПТ на решение конкретного алгоритма или целого класса задач, что позволит наиболее оптимальным способом обеспечить условия эффективной поддержки параллельных вычислений, а именно: равномерно загрузить вычислительный процесс (предупредить простои) и сократить интенсивность взаимодействия вычислительных ядер (независимость). Подобный путь развития РПА представляется наиболее интересным и перспективным, учитывая изначальную ориентированность на «гибридность».

6) Достоинства и недостатки рекуррентной потоковой архитектуры

Ключевые недостатки потоковых архитектур, в результате которых на рынке до сих пор нет коммерчески успешных потоковых процессоров были

перечислены выше. В целом, если рассматривать их глобально, недостатки потоковых архитектур сводятся к двум: необходимость в больших объемах памяти и частое обращение к ним, и специфичность программирования, вызывающая трудности у программистов, как результат приводящая также к плохой совместимости с традиционными системами и программным обеспечением.

В РПА сразу два механизма направлено на сокращение объемов памяти и уменьшение частоты обращения к ней: введение единого потока самодостаточных данных и рекуррентно-свернутое представление алгоритма. Кроме того, за счет рекуррентного представления алгоритма накладные расходы связанные со взаимодействием и передачей информации с управляющего уровня на операционный сведены к минимуму. Разворачивание вычислительного процесса происходит автоматически на аппаратном уровне.

Реализация управляющего уровня в виде традиционного процессора позволяет упростить задачу совместимости с классическими вычислительными системами за счет использования готовых библиотек и IP-ядер. Кроме того, для создания программ, предназначенных для исполнения на процессоре с РПА, при наличии специализированной библиотеки API, можно использовать классические языки программирования и средства разработчика, в том числе распространяемые бесплатно.

Существуют исследования, которые показали что энергия требуемая для обеспечения данных и команд арифметическим устройствам традиционных встраиваемых RISC-процессоров, в 15-20 раз больше энергии, требуемой для выполнения самой команды [6]. Поэтому потенциально можно ожидать еще одно положительное свойство при реализации вычислительной системы с РПА: в РПА на выборку данных требуется значительно меньше энергоемких ресурсов, чем в классических RISC - подобных ЦСП, а выборка команд отсутствует как таковая.

Однако РПА, как и любая другая архитектура не лишена недостатков. Для нее также требуется широкополосная пропускная среда для передачи информации по внутренним шинами. За счет простоты реализации универсального преобразователя тегов, разрядность внутренних шин увеличивается, что потенциально, может привести к различным последствиям, начиная от повышенной теплоотдачи и увеличения энергопотребления, до удорожания всей системы в целом. Впрочем, если оправдается утверждение, указанное в [6], некоторое увеличение энергопотребления за счет увеличения разрядности внутренних шин будет компенсировано «запасом прочности» заложенным в архитектурные особенности РПА.

Также, открытым остается общий для всех разработчиков параллельных систем вопрос о необходимости создания модели программирования, которая вызовет минимальные трудности у программистов при переходе от классического программирования к программированию на процессоре с РПА.

Вывод

Ответы на поднятые вопросы требуют экспериментальной проверки, а выявленные недостатки требуют дополнительных исследований. **В** целом же положительные свойства РПА в условиях кризиса традиционных архитектурных

подходов представляют собой серьезный исследовательский интерес, и при наличии соответствующих ресурсов и продолжении исследований способны привести РПА к определенному успеху, и обеспечить свое место в классификации вычислительных систем.

Литература

1) *Черняк Л.* Архитектура фон Неймана, реконфигурируемые компьютерные системы и антимашина.// Журнал «Открытые Системы». №6 2008
2) *Бурцев ВС.* Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ: Сборник статей / Со-ставители *Торчигин В.П., Никольская Ю.Н., Никитин Ю.В.* - М.: ТОРУС ПРЕСС, 2006. — 416 с.
3) *С.П. Головков, КН. Ефимкин.* Реализация языка программирования для модели вычислений, основанной на принципе потока данных. В сб. Методы и средства обработки информации, Труды Первой Всероссийской научной конференции 'Методы и средства обработки информации', 1-3 октября, Москва, 2003, с. 354-360.
4) *Волчек В.Н.* Современные микропроцессоры и потоковые архитектуры // 1-я Школа молодых ученых ИПИ РАН, Москва, 2010.
5) *В.НВолчек, Степченков Ю.А., Петрухин В.С., А.А. Прокофьев, Р.А. Зеленое.* Цифровой сигнальный процессор с нетрадиционной рекуррентной потоковой архитектурой. // Проблемы разработки перспективных микро- и наноэлектронных систем - 2010. Сборник трудов. - М.: ИППМ РАН,2010.- 694 с,
6) *Кузнецов С.* ИКТ и всемирное развитие. Журнал «Открытые Системы». №6 2008

Сведения об авторе:

Волчек Виктор Николаевич, год рождения: 1985. Место учебы: аспирантура ИПИ РАН. Место работы: ИПИ РАН, 22 отд., инженер- исследователь. Направления научных интересов:

- нетрадиционные архитектуры микропроцессоров;
- параллельные вычисления.